X

Visualization Approaches

X.1 INTRODUCTION

Human-centric text mining emphasizes the centrality of user interactivity to the knowledge discovery process. As a consequence, text mining systems need to provide users with a range of tools for interacting with data. For a wide array of tasks, these tools often rely on very simple graphical approaches such as pick lists, drop-down boxes, and radio boxes that have become typical in many generic software applications to support query construction and the basic browsing of potentially interesting patterns.

In large document collections, however, problems of pattern and feature overabundance have led the designers of text mining systems to move toward the creation of more sophisticated visualization approaches to facilitate user interactivity. Indeed, in document collections of even relatively modest size, tens of thousands of identified concepts and thousands of interesting associations can make browsing with simple visual mechanisms such as pick lists all but unworkable. More sophisticated visualization approaches incorporate graphical tools that rely on advances in many different areas of computer and behavioral science research to promote easier and more intensive and iterative exploration of patterns in textual data.

Many of the more mundane activities that allow a user of a text mining system to engage in rudimentary data exploration are supported by a graphic user interface that serves as the type of basic viewer or *browser* discussed in Chapter IX. A typical basic browsing interface can be seen in Figure X.1.

This type of basic browsing often combines a limited number of query-building functions with an already refined or constrained view of a subset of the textual data in the document collection. In addition, sometimes a basic browsing interface supplements its more character-oriented display elements by supporting the simplified execution of subroutines to draw static graphs of query results.

Text mining visualization approaches, on the other hand, generally emphasize a set of purposes different from those that underpin basic browsing interfaces. Although both basic browsers and visualization tools aim at making interaction with data possible, visualization tools typically result in more sophisticated graphical

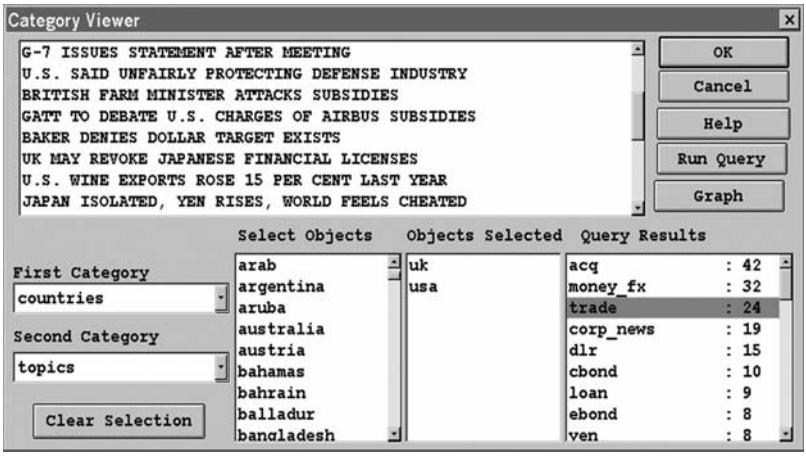

Figure X.1. Basic category browsing in a text mining system. (From Feldman, Kloesgen, Ben-Yehuda, et al. 1997.)

interfaces that attempt to stimulate and exploit the visual capacities of users to identify patterns.

For instance, an interactive circle graph – a common visualization tool in text mining systems – might be tailored specifically to allow cancer researchers to explore an entire corpus of medical research literature broadly in a single graph (see background graph in Figure X.2). By having concepts extracted from the literature represented as nodes on the periphery of the circle and associations between concepts identified by linking lines of various thicknesses that bisect the circle, a researcher could very quickly navigate high-level concepts and then zero-in on relationships emanating from more granular concepts – all while gaining at least a generalized sense of the totality of relationships within the corpus.

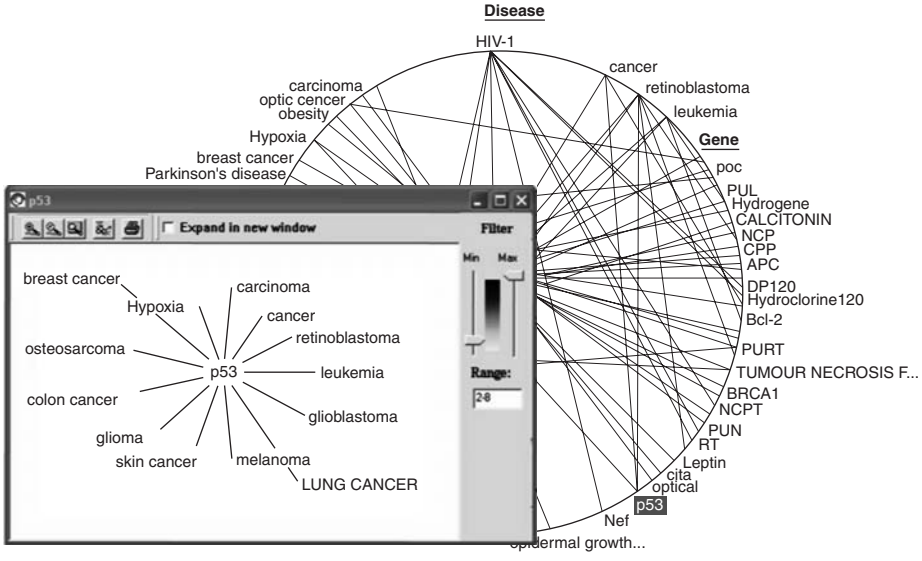

Figure X.2. Circle graph–based category connection map of medical literature relating to AIDS with inset of graphically driven refinement filter. (From Feldman, Regev, et al. 2003.)

This type of visualization tool enables a researcher to appraise, handle, and navigate large amounts of data quickly and with relative ease. Moreover, "control elements" – such as refinement filters or other constraint controls – can be embedded into the overall operations of the visualization interface being executed by as little as a mouse click on a highlighted concept label (see inset in Figure X.2).

Certainly, some kinds of refinement constraints lend themselves to being set quite adequately by character-driven menus or pull-down boxes. By facilitating context-sensitive and graphical refinement of query results, however, more sophisticated visual presentation tools can add to the speed and intuitive ease with which a user can shape knowledge-discovery activities.

Critical advantages that individual visualization approaches can have over character-oriented browsing formats in presenting patterns in data include the following:

- ***Concision:*** the capability of showing large amounts of different types of data all at once;
- ***Relativity and Proximity:*** the ability to easily show clusters, relative sizes of groupings, similarity and dissimilarity of groupings, and outliers among the data in query results;
- ***Focus with Context:*** the ability to interact with some highlighted feature while also being able to see the highlighted feature situated in some of its relational context;
- ***Zoomability:*** the ability to move from micro to macro quickly and easily in one big step or in increments;
- ***"Right Brain" Stimulation:*** the ability to invite user interaction with textual data that is driven not only by premeditated and deliberate search intentions but also as a result of intuitive, reactive, or spatially oriented cognitive processes for identifying interesting patterns.

On the other hand, adding an overabundance of complex graphical features to a visualization interface does not necessarily make the interface more appropriate to its search tasks. Overly complex visualization tools can overdetermine or even inhibit exploration of textual data – particularly if designers of text mining systems lose sight of the main advantages that graphic presentation elements have over more prosaic form- or table-based browser formats.

The evolution from simple, primarily character-based browsers to more powerful and more specialized visualization interfaces has helped transform the orientation of text mining systems. Text mining systems have moved from a focus on the premeditated search for *suspected* patterns to a broader capability that also includes more free-form and unguided exploration of textual data for implicit, obscure, and *unsuspected* patterns.

X.1.1 Citations and Notes

A seminal discussion of human-centered knowledge discovery can be found in Brachman and Anand (1996). Grinstein (1996) also offers a relevant treatment of related topics.

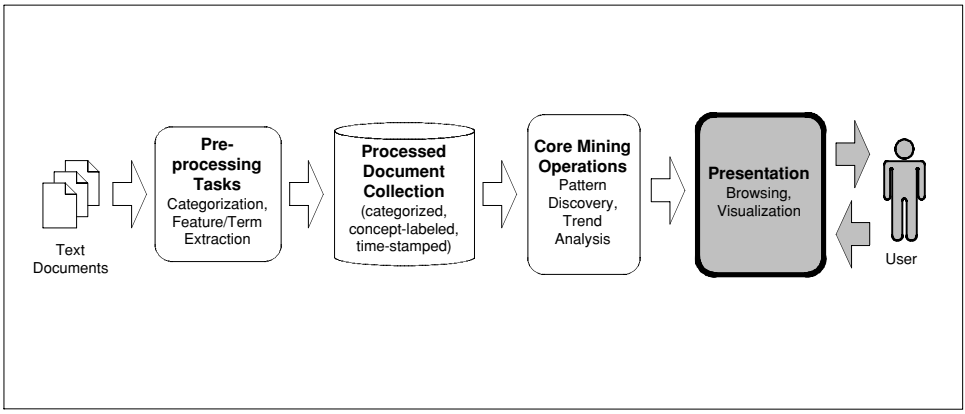

Figure X.3. High-level functional architecture of a text mining system showing position of visualization.

General overviews of information visualization can be found in Tufte (1983), Tufte (1990), Cleveland (1994), Shneiderman (1997), and Spence (2001). Useful works on information visualization techniques in information retrieval and the visual presentation of query results include Rao et al. (1992), Spoerri (1999), Ahlberg and Schneiderman (1994), Masui et al. (1995), Hearst (1999), Lagus (2000b), and Hearst (2003). Important early treatments of information navigation and exploration approaches include Goldstein and Roth (1994), Ahlberg and Wistrand (1995), and Jerding and Stasko (1995).

There really is not yet a comprehensive treatment of visualization techniques specific to text mining. However, several works – including Feldman, Kloesgen, Ben-Yehuda, et al. (1997); Feldman, Kloesgen, and Zilberstein (1997a); Landau et al. (1998); Aumann et al. (1999); Lagus et al. (1999); Wong, Whitney, and Thomas (1999); Lagus (2000a); and Wong et al. (2000) – provide relevant discussions of a limited number of specific visual techniques and their application to text mining activities.

X.2 ARCHITECTURAL CONSIDERATIONS

In the high-level functional architecture of a text mining system illustrated in Figure X.3, visualization tools are among those system elements situated closest to the user. Visualization tools are mechanisms that serve to facilitate human interactivity with a text mining system. These tools are layered on top of – and are dependent upon – the existence of a processed document collection and the various algorithms that make up a text mining system's core mining capabilities.

The increased emphasis on adding more sophisticated and varied visualization tools to text mining systems has had several implications for these systems' architectural design. Although older text mining systems often had rigidly integrated visualization tools built into their user interface (UI) front ends, newer text mining systems emphasize modularity and abstraction between their front-end (i.e., presentation layer) and middle-tier (i.e., core discovery and query execution elements) architectures.

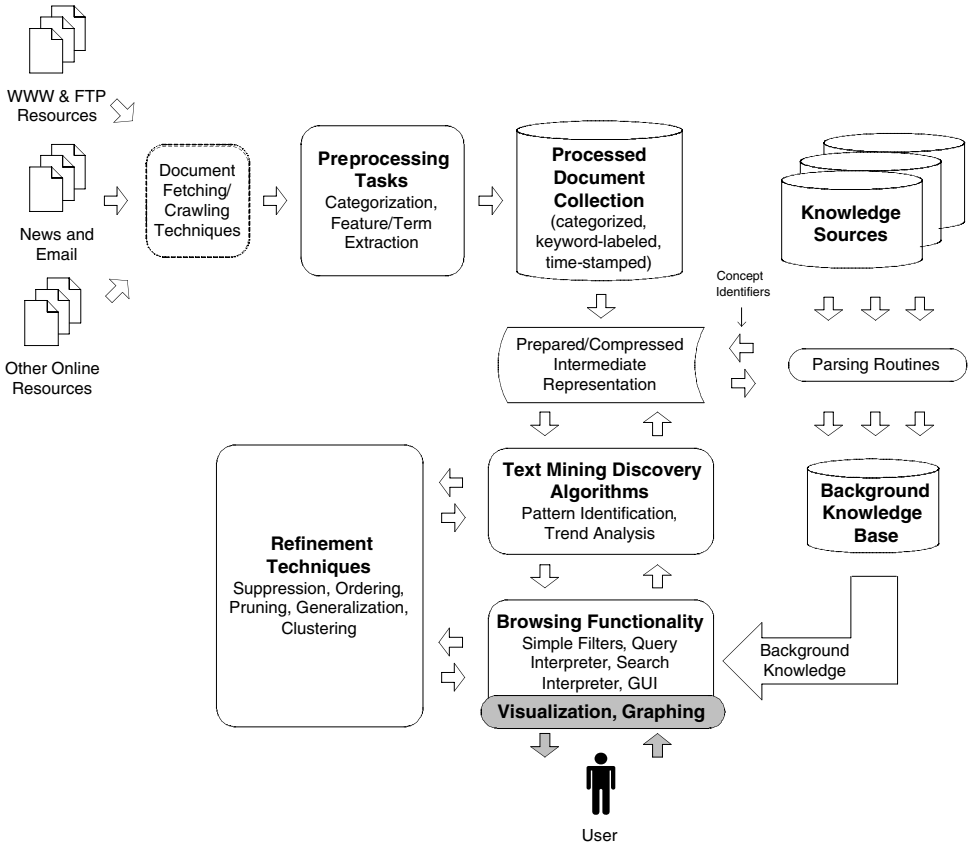

Figure X.4. Situating visualization within text mining system architecture.

Indeed, there are several good reasons for architects of text mining systems to abstract the front and middle tiers of their software platforms. First, visualization tools and knowledge discovery algorithms tend to be modified and upgraded on an ever more iterative basis. A "decoupled" or "loosely coupled" front end and middle tier in text mining system – abstracted from each other by an intermediary connection layer based on a formal and well-defined software interface – allow much better for such unsynchronized development of different elements of the text mining system. Figure X.4 illustrates the general position of visualization components in a text mining system's architecture.

Second, text mining systems are moving from having a few limited visualization and graphing tools to supporting whole suites of different kinds of presentation layer utilities. This is both a reflection of the movement toward facilitating greater user interactivity through more customized (even personalized) UIs and the proliferation of more mature, sophisticated, and specialized visualization tools. With many more types of different visualization approaches now available, architects of text mining systems are probably well advised to keep their options open; instead of scrapping a whole text mining system when its UI has become hopelessly outdated, developers can leverage a more loosely coupled front end and middle-tier architecture to continue to add additional visualization components.

Finally, from a practical perspective, the wider availability of RDF and XML-oriented protocols makes such loose coupling of front ends and middle tiers much more feasible. This fact is underscored by the current availability of a whole spate of specialized and very powerful commercial off-the-shelf visualization software with defined interfaces or feed formats that support various RDF or XML data interchange approaches.

Visualization tools have increasingly played a crucial, even transformative role in current state-of-the-art text mining systems. As with data mining systems, sophisticated visualization tools have become more critical components of text mining applications because of their utility in facilitating the exploration for hidden and subtle patterns in data.

X.2.1 Citations and Notes

For obvious reasons, the architectural discussion in this section is highly generalized. The architectural descriptions have been informed by the visualization elements found in the KDT (Feldman and Dagan 1995), Explora (Kloesgen 1995b), Document Explorer (Feldman, Kloesgen, and Zilberstein 1997a), and TextVis (Landau et al. 1998) knowledge discovery systems.

X.3 COMMON VISUALIZATION APPROACHES FOR TEXT MINING

X.3.1 Overview

A substantial and mature literature already exists relating to the use of visualization tools in a wide range of generic and specific computer science applications. The aim of the next few sections is to illustrate how a select number of commonly seen visualization approaches have been put to good use supplementing text mining functionality.

The potential number of combinations of visual techniques that can be applied to problems in unstructured data is probably limitless. With such a wide array of possible visual techniques, coupled with the subjective nature of assessing the efficacy of visualization approaches across different types of knowledge-discovery problem sets and user groups, it would be problematic to attempt to rate, to any precise degree, the success of a specific visual approach or set of approaches. Nevertheless, several visual approaches have suggested themselves more informally as useful enhancements to knowledge discovery operations involving textual data. These include simple concept graphs, histograms, line graphs, circle graphs, self-organizing maps, and so-called context + focus approaches – like the hyperbolic tree – as well as various derivative and hybrid forms of these main approaches.

Thus, perhaps it should be stated clearly that the intention here is not so much to be *prescriptive* – detailing the circumstances when a particular visualization approach is decidedly more appropriate, more powerful, or more effective than another for a given task – as *descriptive*, or describing how a particular tool has typically been employed to supplement text mining systems.

X.3.2 Simple Concept Graphs

Even rather bare-bones visualization tools such as *simple concept graphs* provide an efficient exploration tool for getting familiar with a document collection. The two main benefits of these types of visualizations are their abilities to *organize* the exploration of textual data and to *facilitate interactivity* – that is, the user can click on each node or edge and get the documents supporting them or can initiate various other operations on the graphs. There is a relationship between these two benefits as well: offering user-friendly organization approaches can do much to promote increased user interactivity with textual data.

This latter type of exploration can be further supported by linking several graphs. Thus, the relevance of selected aspects of one graph can be efficiently studied in the context of another graph. Simple concept graphs have been used, with many variations, in several real-world text mining systems.

Simple Concept Set Graphs

One of the most basic and universally useful visualization tools in text mining is the simple "root and branches" hierarchical tree structure. Figure X.5 shows a classic visualization for a concept taxonomy in a document collection. The root and leaf vertices (nodes) of such a visualization are concept identifiers (i.e., name labels for concepts). The special layout of the presentation elements allows a user to traverse the hierarchical relationships in the taxonomy easily either to identify sought-after concepts or to search more loosely for unexpected concepts that appear linked to other interesting concepts in the hierarchy.

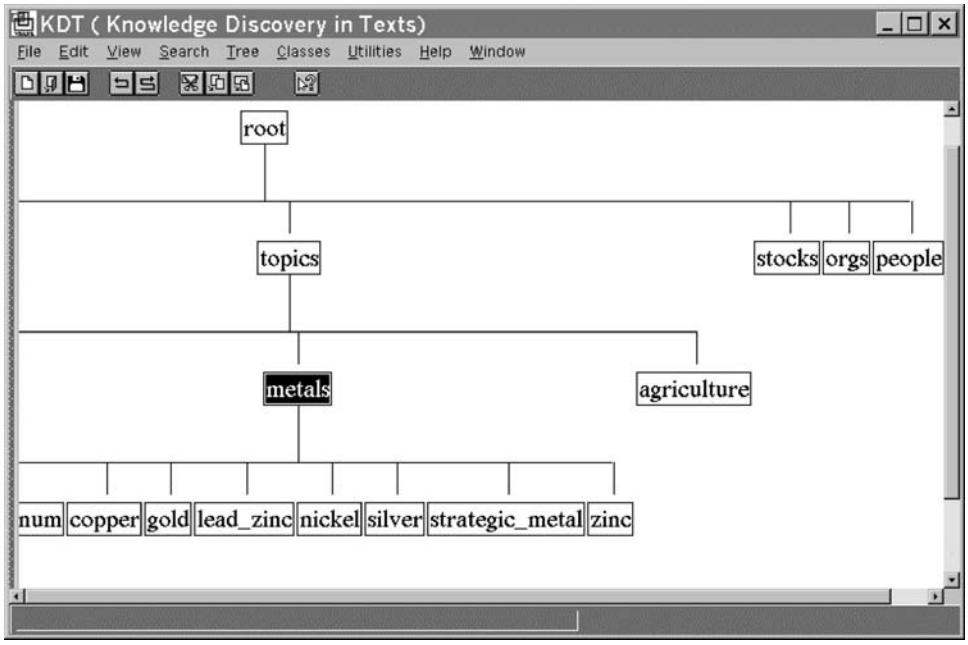

Figure X.5. Interactive graph used to illustrate a concept taxonomy as a hierarchical tree structure. (From Feldman, Dagan, and Hirsh 1998. Reprinted with permission of Springer Science and Business Media.)

This kind of visualization tool can also easily be made to allow a user to click on a node concept and either move to the underlying documents containing the concept or to connect to information about sets or distributions of documents containing the concept within the document collection. This latter type of information – the answer set to a rather routine type of query in many text mining systems – can be demonstrated by means of a concept set graph.

Formally, a *concept set graph* refers to a visual display of a subset of concept sets with respect to their partial ordering. Perhaps the most common and straightforward way to display concept sets graphically is also by means of a simple hierarchical tree structure.

Figure X.5 shows a set graph for frequent sets arranged in a tree structure. The user can operate on this graph by selecting nodes, opening and closing nodes, or defining new search tasks with respect to these nodes, for instance, to expand the tree.

The first level in Figure X.5 relates to country concepts sorted by a simple quality measure (support of the frequent set). The node "USA" (support: 12,814 documents) is expanded by person concepts. Further expansions relate to economical topic concepts (e.g., expansion of the node "James Baker": 124 documents, 0%) and country concepts.

Of course, a hybrid form could be made between the "root and branches"–type visual display format shown in Figure X.5 and the simple concept set graph illustrated in Figure X.6. For some applications, having the percentage support displayed within a concept node on the root and branches visualization might prove more useful to

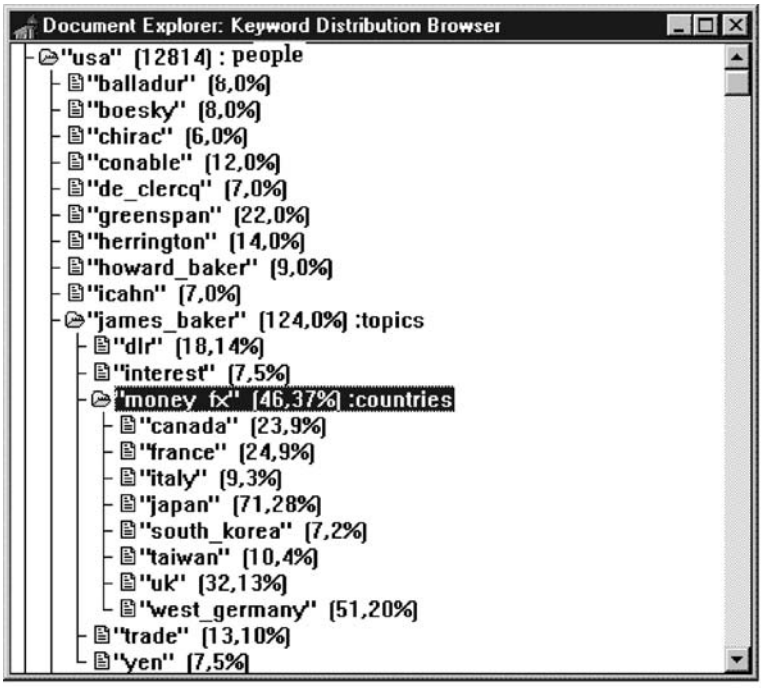

Figure X.6. A hierarchical concept set graph. (From Feldman, Kloesgen, and Zilberstein 1997b.)

navigation and exploration than the "long indented list" appearance of the vertical tree structure in Figure X.6. This form would be a directed graph, the edges of which (usually depicted with directional arrow heads) indicate the hierarchical relationship between nodes at the graph vertices.

Although a hierarchical concept graph may represent a very basic approach to visualizing sets of concepts, it can be also used as the entry point for jumping to more complex visualizations or graphically driven refinement controls. For instance, by clicking on a concept identifier, a user may be able to navigate to another graph that shows associations containing the highlighted concept, or a graphic box could be triggered by clicking on a concept allowing the user to adjust the quality measure that drove the original query.

Another related, commonly used visualization approach applicable to simple concept sets is the organization of set members into a DAG. Formally, a DAG can be described as a directed graph that has no path and that begins and ends at the same vertex. Practically, it might be viewed as a hierarchical form in which child nodes can have more than one parent node.

DAGs can be useful in describing more complex containership relations than those represented by a strict hierarchical form, in that the DAG represents a generalization of a tree structure in which a given subtree can be shared by different parts of the tree. For instance, a DAG is often used to illustrate the somewhat more complex relations between concepts in an ontology that models a real-world relationship set in which "higher level" concepts often have multiple, common directed relationships with "lower level" concepts in the graph. Described in a different way, DAGs permit lower level containers to be "contained" within more than one higher level container at the same time.

A very simple DAG is shown in Figure X.7. Traditional, rigidly directed hierarchical representations might be both much less obvious and less efficient in showing that four separate concepts have a similar or possibly analogous relationship to a fifth concept (e.g., the relationship that the concepts *motor vehicles*, *cars*, *trucks*, and *power tillers* have with the concept *engines*). Because of their ability to illustrate more complex relationships, DAGs are very frequently leveraged as the basis for moderately sophisticated relationship maps in text mining applications.

A more complex and well-known application of a DAG to an ontology can be seen in Zhou and Cui's (Zhou and Cui 2004) visual representations of the Gene Ontology (GO) database. Zhou and Cui created a DAG to visually model a small subset of the GO ontology, focusing only on the root node and three child nodes. When using a DAG to show biological function for 23 query genes, the DAG still ended up having 10 levels and more than 101 nodes.

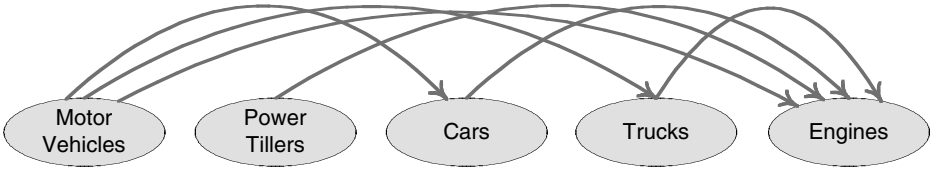

Figure X.7. A simple DAG modeling a taxonomy that includes multiple parent concepts for a single-child concept.

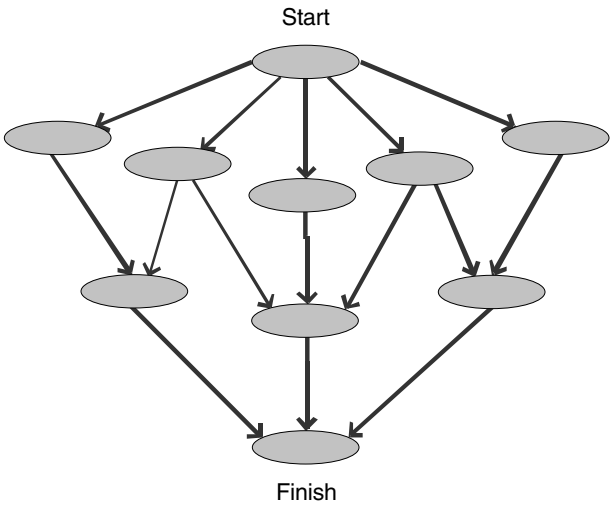

Figure X.8. Visualization of a generic DAG-based activity network.

Visualization techniques based on DAGs have proven very useful for visually modeling complex set-oriented or container-type relationships, such as those found among concepts in the GO ontology, in a relatively straightforward and understandable way. However, Zhou and Cui find DAGs to be limited when illustrating more granular or functional relationship information (such as one finds when exploring concept associations) or when the number of concepts in play becomes large. In these cases, other types of visualization techniques, such as circular graphs or network models, can sometimes provide greater expressive capabilities.

Beyond their use in modeling concept hierarchies, DAGs can also be employed as the basis for modeling *activity networks*. An activity network is a visual structure in which each vertex represents a task to be completed or a choice to be made and the directed edges refer to subsequent tasks or choices. See Figure X.8.

Such networks provide the foundation for more advanced types of text mining knowledge discovery operations. DAG-based activity networks, for instance, form the basis for some of the more popular types of visualizations used in *critical path analysis* – often an important approach in knowledge-discovery operations aimed at link detection.

Simple Concept Association Graphs

Simple concept association graphs focus on representing associations. A simple association graph consists of *singleton vertex* and *multivertex graphs* in which the edges can connect to a set of several concepts. Typically, a simple association graph connects concepts of a selected category. At each vertex of a simple association graph, there is only one concept. Two concepts are connected by an edge if their similarity with respect to a similarity function is larger than a given threshold.

A simple concept association graph can be undirected or directed, although undirected graphs are probably more typical. For example, one might use an undirected graph to model associations visually between generic concepts in a document collection generated from corporate finance documentation. On the other hand, if one

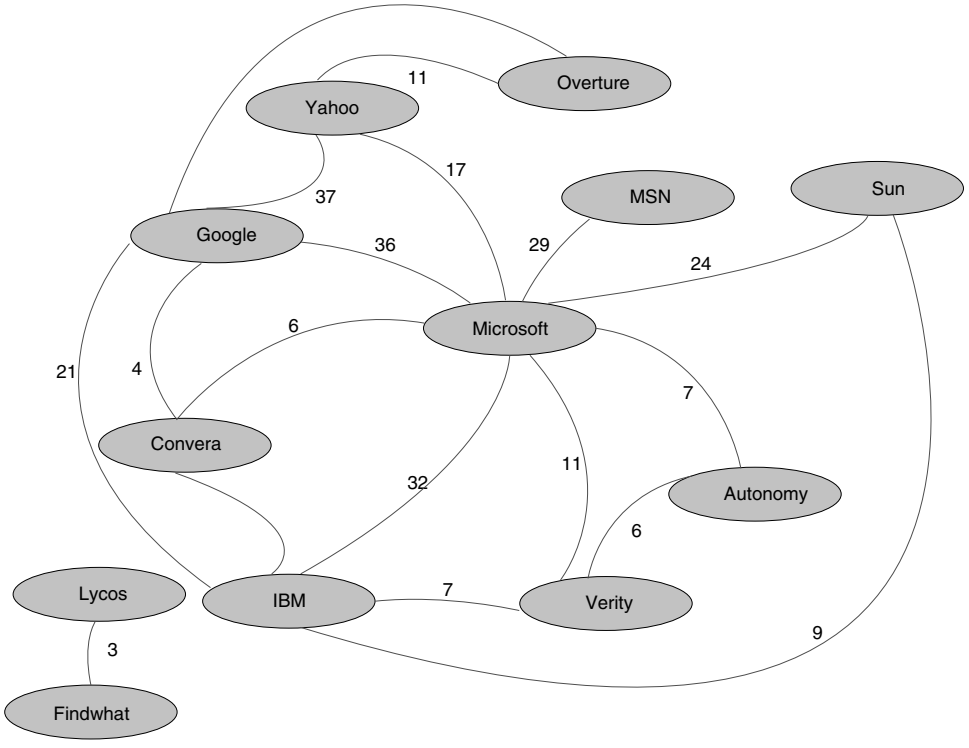

Figure X.9. Concept association graph: single vertex, single category (software companies in the context of search engine software).

were seeking to produce a tool to visualize associations between proteins in a corpus of proteomics research literature, one might want to employ a directed graph with directed edges (as denoted by directional arrowheads) between concept nodes. This type of directed graph would be useful in visually indicating not just general association relationships but also the patterns of one protein's acting upon another.

Figure X.9 shows a concept association graph for the *company* category in the context of *search engine software* and a simple similarity function based on the number of documents in which the companies co-occur. The figure allows a user to quickly infer conclusions about data that might be possible only after a much more careful investigation if that user were forced to make his or her way through large lists or tables of textual and statistical data. Such inferences might include the following:

- Microsft, Google, and IBM are the most connected companies;
- Lycos and Findwhat are the only members of a separate component of the graph;
- MSN is connected only to Microsoft, and so on.

Another type of simple concept association graph can present the associations between different categories such as companies and software topics. The singleton vertex version of this graph is arranged like a map on which different positions of circles are used to include the concepts of categories, and edges (between companies and software topics) present the associations. Often, these singleton vertex graphs

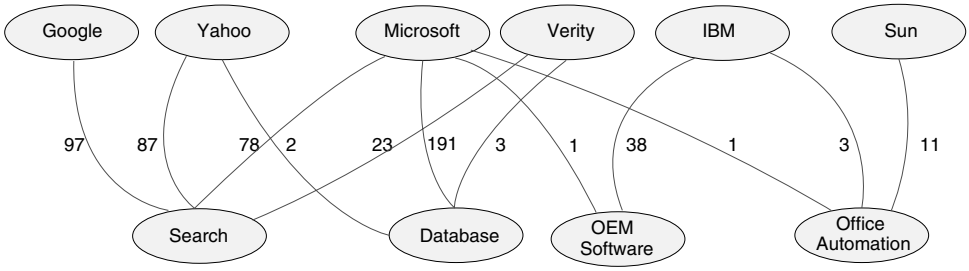

Figure X.10. Concept association graph: single vertex, several categories.

are designed as bipartite graphs displaying two categories of concepts by splitting one category to the top of the graph and another category to the bottom with edges representing connections linking individual pairs of vertices. Figure X.10 shows an example of this kind of concept association graph.

Similarity Functions for Simple Concept Association Graphs

Similarity functions often form an essential part of working with simple concept association graphs, allowing a user to view relations between concepts according to differing weighting measures. Association rules involving sets (or concepts) A and B that have been described in detail in Chapter II are often introduced into a graph format in an undirected way and specified by a support and a confidence threshold. A fixed confidence threshold is often not very reasonable because it is independent of the support from the RHS of the rule. As a result, an association should have a significantly higher confidence than the share of the RHS in the whole context to be considered as interesting. Significance is measured by a statistical test (e.g., t-test or chi-square).

With this addition, the relation given by an association rule is undirected. An association between two sets A and B in the direction $A \Rightarrow B$ implies also the association $B \Rightarrow A$. This equivalence can be explained by the fact that the construct of a statistically significant association is different from implication (which might be suggested by the notation $A \Rightarrow B$). It can easily be derived that if B is overproportionally represented in A, then A is also overproportionally represented in B.

As an example of differences of similarity functions, one can compare the undirected connection graphs given by statistically significant association rules with the graphs based on the cosine function. The latter relies on the cosine of two vectors and is efficiently applied for continuous, ordinal, and also binary attributes. In case of documents and concept sets, a binary vector is associated to a concept set with the vector elements corresponding to documents. An element holds the value 1 if all the concepts of the set appear in the document. Table X.1 (Feldman, Kloesgen, and Zilberstein 1997b), which offers a quick summary of some common similarity functions, shows that the cosine similarity function in this binary case reduces to the fraction built by the support of the union of the two concept sets and the geometrical mean of the support of the two sets.

A connection between two sets of concepts is related to a threshold for the cosine similarity (e.g., 10%). This means that the two concept sets are connected if the support of the document subset that holds all the concepts of both sets is larger than 10 percent of the geometrical mean of the support values of the two concept sets.

Table X.1. *Some Commonly Used Similarity Functions for Two Concept Sets A, B (a = support(A), b = support(B), d = support(A,B))*

Function	Similarity	Characteristic
Support threshold	$d > d_0$ (step function)	evaluates only d, independent from $a - d$, $b - d$
Cosine	$s = d/\sqrt{a \cdot b}$	Low weight of $a - d$, $b - d$
Arithmetical mean	$s = 2d/(a + b)$	middle point between cosine and Tanimoto
Tanimoto	$s = d/(a + b - d)$	high weight of $a - d$, $b - d$
Information measure	weighted documents	only applicable if weights are reasonable
Statistical test	threshold statist. quality	typically for larger samples and covers

The threshold holds a property of monotony: If it is increased, some connections existing for a lower threshold disappear, but no new connections are established. This property is used as one technique to tune the complexity of a simple concept graph.

One can derive a significance measure (factor f) for this situation in which tuning is required in the following way.

Let f be the following factor:

$$f = Ns(A, B)/s(A)s(B).$$

Given the support s for the two concept sets A resp. B and N the number of documents in the collection (or a subcollection given by a selected context), we can calculate the factor. In the case of the independence of the two concept sets, f would be expected around the value 1. Thus, f is larger than 1 for a statistically significant association rule.

The cosine similarity of concept sets A and B can now be calculated as

$$S(A, B) = f \cdot \sqrt{q(A) \cdot q(B)};$$

that is, as the geometrical mean of the relative supports of A and B ($q(A) = s(A)/N$) multiplied by the factor f, thus combining a measure for the relative support of the two sets (geometrical mean) with a significance measure (factor f).

The cosine similarity therefore favors connections between concept sets with a large support (which need not necessarily hold a significant overlapping) and includes connections for concept sets with a small support only if the rule significance is high. This means that the user should select the cosine similarity option if there is a preference for connections between concept sets with a larger support.

On the other side, the statistically based association rule connection should be preferred if the degree of coincidence of the concepts has a higher weight for the analysis. Similar criteria for selecting an appropriate similarity function from Table X.1 can be derived for the other options.

Equivalence Classes, Partial Orderings, Redundancy Filters

Very many pairs of subsets can be built from a given category of concepts, (e.g., all pairs of country subsets for the set of all countries). Each of these pairs is a possible association between subsets of concepts. Even if the threshold of the similarity

function is increased, the resulting graph can have too complex a structure. We now define several equivalence relations to build equivalence classes of associations. Only a representative association from each class will then be included in the keyword graph in the default case.

A first equivalence is called *cover equivalence*. Two associations are cover-equivalent iff they have the same cover. For example (Iran, Iraq) => (Kuwait, USA) is equivalent to (Iran, Iraq, Kuwait) => USA because they both have the same cover (Iran, Iraq, Kuwait, USA). The association with the highest similarity is selected as the representative from a cover equivalence class.

Context equivalence is a next equivalence relation. Two associations are context-equivalent iff they are identical up to a different context. That means that two associations are identical when those concepts that appear on both sides are eliminated from each association. For example, (Iran, Iraq) => (Iran, USA) is equivalent to (Kuwait, Iraq) => (Kuwait, USA). The first association establishes a connection between Iraq and USA in the context of Iran, whereas the second association is related to the context of Kuwait. The context-free associations are selected as the representatives from this equivalence class (e.g., Iraq => USA).

The next definition relates to a partial ordering of associations, not an equivalence relation. An association A1 is stronger than an association A2 if the cover of A1 is a subset of the cover of A2. As special cases of this ordering, the right- and left-hand sides are treated separately.

Selecting the representative of an equivalence class or the strongest associations can be applied as a basic redundancy filter. Additionally, criteria can refine these filters (for instance, for the context-equivalence, a context-conditioned association can be selected in addition to the context-free association iff the similarity of the context-conditioned association is much higher with respect to a significance criterion).

There is a duality between frequent sets of concepts and associations. For a given set of frequent concepts, the implied set of all associations between frequent concepts of the set can be introduced. On the other hand, for a given set of associations, the set of all frequent concepts appearing as left- or right-hand sides in the associations can be implied.

In the application area of document collections, users are mainly interested in frequent concept sets when concentrating on basic retrieval or browsing. These frequent concepts are considered as retrieval queries that are discovered by the system to be interesting.

When attempting to gain some knowledge of the domain represented by a document collection, users are often drawn to interacting with association rules, shaping the various measures and refinement filters to explore the nature of the concept relations in the domain. In the simple concept graphs, the concept sets are therefore included as active nodes (activating a query to the collection when selected by the user). Complementary and intersection sets (e.g., related to the cover of an association) can also appear as active nodes.

Typical Interactive Operations Using Simple Concept Graphs

One of the key drivers for employing visualization tools is to promote end user interactivity. Concept graphs derive much of their value from facilitating interactive

operations. A user can initiate these operations by manipulating elements in the graphs that execute certain types of system activities.

Some interactive operations relating the concept graphs have already been discussed – or at least suggested – in the previous sections. However, a more systematic review of several types of these operations provides useful insights into kinds of secondary functionality that can be supplemented by simple concept graph visualization approaches.

Browsing-Support Operations

Browsing-support operations enable access to the underlying document collections from the concept set visual interface. Essentially, a concept set corresponds to a query that can be forwarded to the collection retrieving those documents (or their titles as a first summary information), which include all the concepts of the set.

Therefore, each concept set appearing in a graph can be activated for browsing purposes. Moreover, derived sets based on set operations (e.g., difference and intersection) can be activated for retrieval.

Search Operations

Search operations define new search tasks related to nodes or associations selected in the graph. A graph presents the results of a (former) search task and thus puts together sets of concepts or sets of associations. In a GUI, the user can specify the search constraints: syntactical, background, quality, and redundancy constraints.

The former search is now to be refined by a selection of reference sets or associations in the result graph. Some of the search constraints may be modified in the GUI for the scheduled refined search.

In refinement operations, the user can, for example, increase the number of elements that are allowed in a concept set. For instance, selected concept sets in Figure X.6 or selected associations in Figure X.9 can be expanded by modifying restrictions on the maximum number of elements in concept sets.

Link Operations

Link operations combine several concept graphs. Elements in one graph are selected and corresponding elements are highlighted in the second graph. Three types of linked graphs can be distinguished: links between set graphs, between association graphs, and between set and association graphs.

When linking two set graphs, one or several sets are selected in one graph and corresponding sets are highlighted in the second graph. A correspondence for sets can rely, for instance, on the intersections of a selected set with the sets in the other graph. Then all those sets that have a high overlap with a selected set in the first graph are highlighted in the second graph.

When selected elements in a set graph are linked with an association graph, associations in the second graph that have a high overlap with a selected set are highlighted. For instance, in a company graph, all country nodes that have a high intersection with a selected topic in an economical topic graph can be highlighted.

Thus, linkage of graphs relies on the construct of a *correspondence* between two set or association patterns. For example, a correspondence between two sets can be defined by a criterion referring to their intersection, a correspondence between a set

and an association by a specialization condition for the more special association constructed by adding the set to the original association, and a correspondence between two associations by a specialization condition for an association constructed by combining the two associations.

Presentation Operations

A first interaction class relates to diverse *presentation options* for the graphs. It includes a number of operations essential to the customization, personalization, calibration, and administration of presentation-layer elements, including

- sorting (e.g., different aspects of quality measures)
- expanding or collapsing
- filtering or finding
- zooming or unzooming nodes, edges, or graph regions.

Although all these presentation-layer operations can have important effects on facilitating usability – and as a result – increased user interaction, some can in certain situations have a very substantial impact on the overall power of systems visualization tools.

Zoom operations, in particular, can add significant capabilities to otherwise very simple concept graphs. For instance, by allowing a user to zoom automatically to a predetermined focal point in a graph (e.g., some concept set or association that falls within a particular refinement constraint), one can add at least something reminiscent of the type of functionality found in much more sophisticated fisheye and self-ordering map (SOM) visualizations.

Drawbacks of Simple Concept Graphs

A few disadvantages of simple concept graphs should be mentioned. First, the functionality and usability of simple concept graphs often become more awkward and limited with high levels of dimensionality in the data driving the models. Hierarchies with vast numbers of nodes and overabundant multiple-parent-noded relationships can be difficult to render graphically in a form that is legible – let alone actionable – by the user of a text mining system; moreover, undirected concept association graphs can become just as intractable for use with large node counts.

In addition, although the streamlined nature of simple concept graphs has its benefits, there are some built-in limitations to interactivity in the very structure of the graph formats. Clearly, tree diagrams can have "hotspots" that allow linking to other graphs, and the type of linked circle node graphs that constitute most simple concept-association graphs can be made to support pruning and other types of refinement operations. However, both forms of simple concept graphs are still relatively rigid formats that are more useful in smaller plots with limited numbers of nodes that can be visually traversed to understand patterns.

As a result, simple concept graphs are far less flexible in supporting the exploration of complex relationships than some other types of visualization approaches. Indeed, other approaches, such as some three-dimensional paradigms and visual methodologies involving greater emphasis of context + focus functionality have been

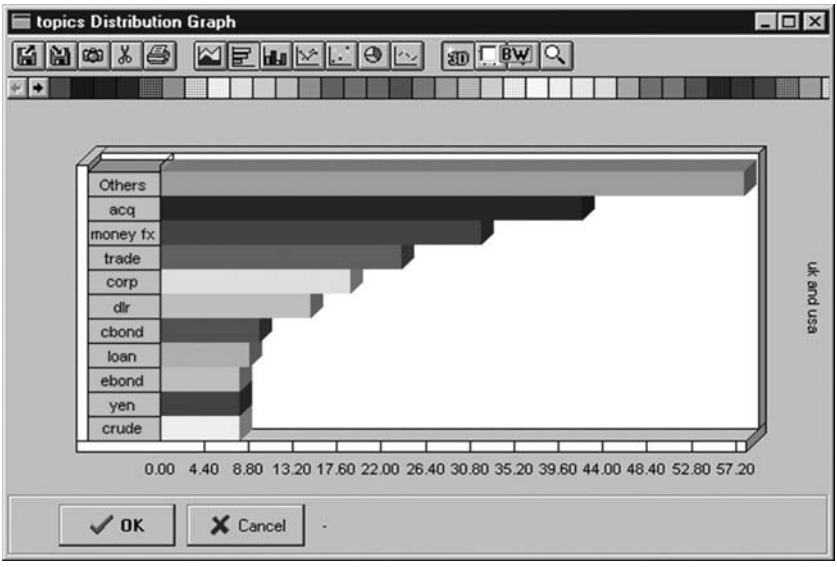

Figure X.11. Early text mining visualization implementation based on a histogram (topic distribution graph from the KDT system ca. 1998). (From Feldman, Kloesgen, and Zilberstein 1997b.)

specifically designed to offer greater flexibility in navigation and exploration of query results than have data with more abundant or complex patterns.

X.3.3 Histograms

In addition to basic "bread and butter" approaches like simple concept graphs, early text mining systems often relied on classic graphing formats such as *histograms*, or bar charts, to provide visualization capabilities. Although architects of text mining systems have shown an increasing willingness to integrate more exotic and complex interactive graphic tools into their systems, histograms still have their uses in exploring patterns in textual data. A very simple histogramatic representation can be seen in Figure X.11.

Histograms still have their uses in text mining systems today and seem particularly well-suited to the display of query results relating to distributions and proportions. However, although two-dimensional (2-D) bar charts themselves have changed little over the last several years, the overall presentation framework in which these bar charts are displayed has become substantially more refined.

Histogrammatic representations are often situated in GUIs with split screens, which also simultaneously display corresponding lists or tables of concept distribution and proportion information of the type described in Chapter II. Histograms are useful in presentation of data related to distributions and proportions because they allow easy comparison of different individual concepts or sets across a wider range of other concepts or sets found within a document collection or subcollection. See Figure X.12. This is not, however, to say that histograms are only useful in displaying results for distribution- or proportion-type queries; for instance, associations can

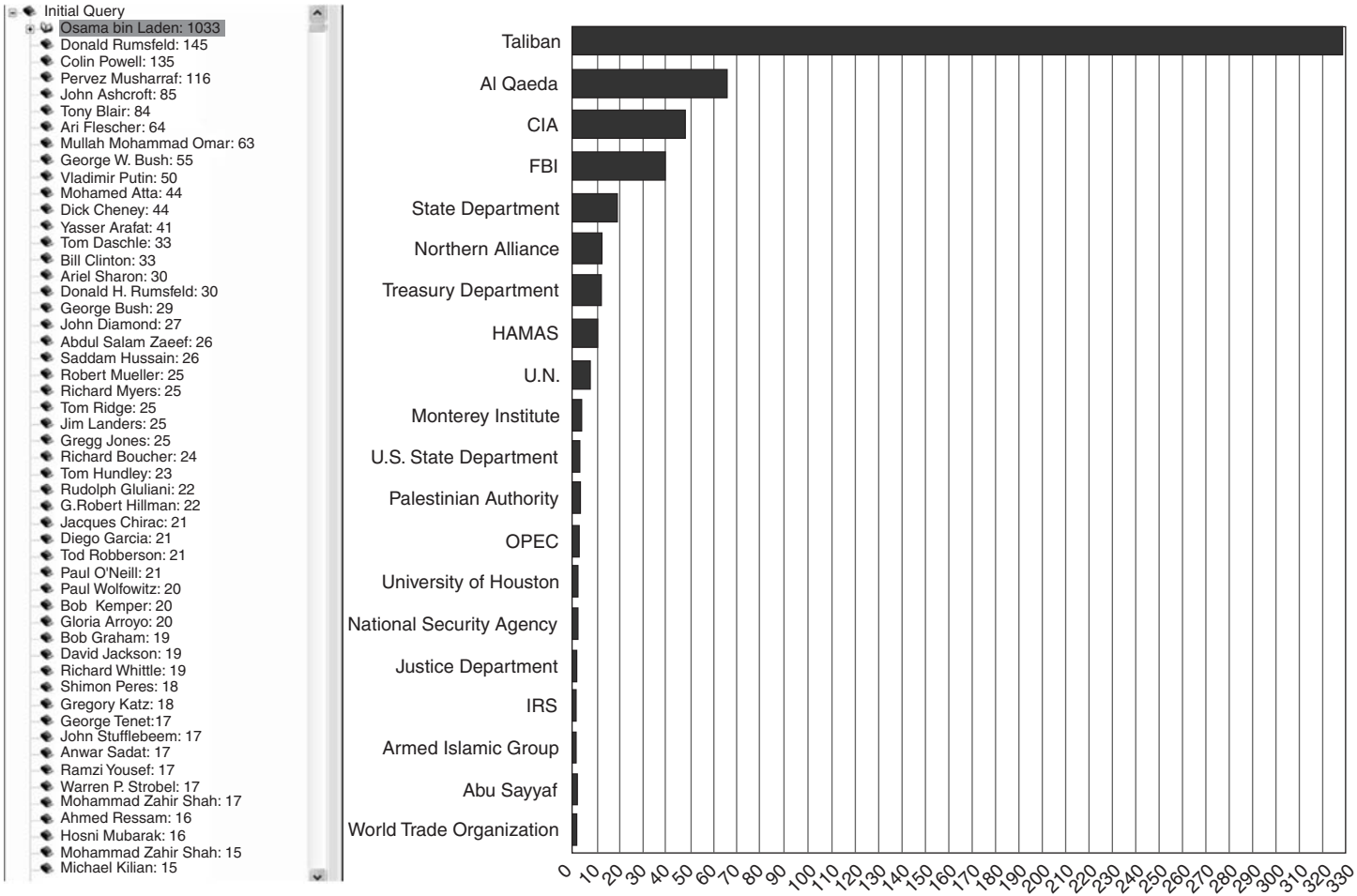

Figure X.12. GUI's left pane shows results of concept distribution query in list form and right pane with histogrammatic representation of this same distribution.

also be compared according to various measures and plotted in stacks. It is true, however, that histograms are more commonly employed to display distribution- and proportion-related results.

Users can thus very easily sort or traverse these lists of concepts (typically along with details pertaining to some measure of quality) while quickly scanning the bar charts for visual cues that suggest relative comparisons of distribution or proportion information, outliers, concepts closer to the average distribution, and so on.

Indeed, histograms now tend to be more interactive inasmuch as users are able to manipulate refinement constraints. The incorporation of pop-ups or separate windows allow filters to be adjusted by means of sliders, dials, buttons, or pull-down boxes to give users a more "real-time" feel for how constraint changes affect query results.

Radical fluctuations in the height of individual bars on a chart are visually much more immediately noticeable to a user than the changes in numerical values in long lists or table grids. Still, smaller differences between bars are harder to discern. Thus, histograms can sometimes be more useful for examining outliers and baseline values among charted items. They are less useful for helping a user visually distinguish between more minute differences in the relative values of individually graphed items.

X.3.4 Line Graphs

Like histograms, line graphs may not at first seem to be the most advanced of visualization approaches for text mining applications. However, these graphs have many advantages. Many academic and commercial text mining systems have at one time or other employed line graphs to support knowledge discovery operations.

Line graphs represent what might be described as "cheap and cheerful" visualization solutions for text mining. They are "cheap" because they combine the virtues of relatively low system overhead and development expense in that there are many widely available free or low-cost line graphing software libraries that can be leveraged to create specific competent presentation elements. They are "cheerful" because many of these mature, prebuilt libraries have been specifically developed to be embedded into a wide range of software applications. As a result, integration and customization of the libraries are often relatively straightforward.

These advantages make line graphs a good choice for developing uncomplicated graphs during the prototyping and early-release stages of text mining systems. The ease of implementing such graphs is helpful because it permits very quick feedback to system developers and users about the performance of text mining algorithms.

Beyond their use as prototyping tools, line graphs have been employed to provide visualizations for numerous tasks relating to a wide array of text mining operations. Two types of visualization approaches relying on line graphs are particularly common.

The first approach involves comparisons across a range of items. By using one axis of the graph to show some measure and the other to itemize elements for comparison, line graphs have been applied to three common analysis techniques:

- Comparisons of the results of different sets of queries,
- Comparisons of a set of common queries run against different document subsets, and

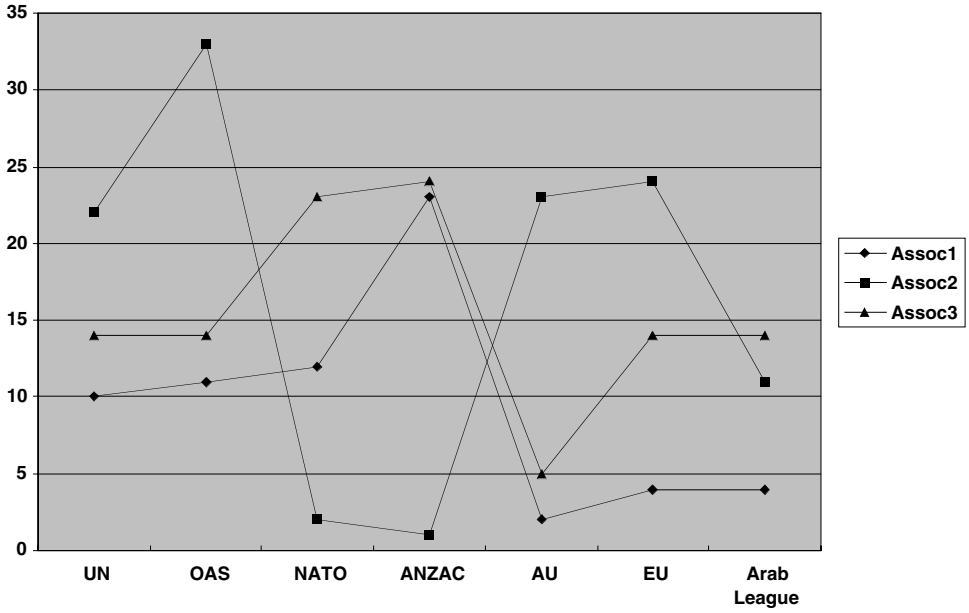

Figure X.13. Line graph showing number of associations for three sets of queries.

- Comparisons of the numbers of concepts that appear under different constraint or quality-measure conditions.

Figure X.13 illustrates the first of these three common techniques in which a line graph displays a comparison of the number of associations for two sets of queries. Note the use of the two axes and the plotting of two distinct lines with different symbols (squares and diamonds) identifying data points.

The second and arguably most prevalent current use of line graphs in text mining is that of graphs displaying trends or quantities over time. Line charts provide a very easily understood graphical treatment for periodicity-oriented analytics with the vertical axis showing quantity levels and the horizontal axis identifying time periods. See Figure X.14.

Line graphs can also be used in hybrids of these two approaches. Using multiline graphs, one can compare various types common to text mining tasks in the context of the time dimension. See example in Figure X.15.

Such applications of line graphs benefit from their concision, for a large amount of information can be displayed simultaneously with clarity. Line graphs, however, may not be the most appropriate visualization modality when a text mining analytical task calls for inviting more immediate interactivity from a user.

X.3.5 Circle Graphs

A circle graph is a visualization approach that can be used to accommodate a large amount of information in a two-dimensional format. It has been referred to as an "at-a-glance" visualization approach because no navigation is required to provide a complete and extremely concise visualization for potentially large volumes of data.

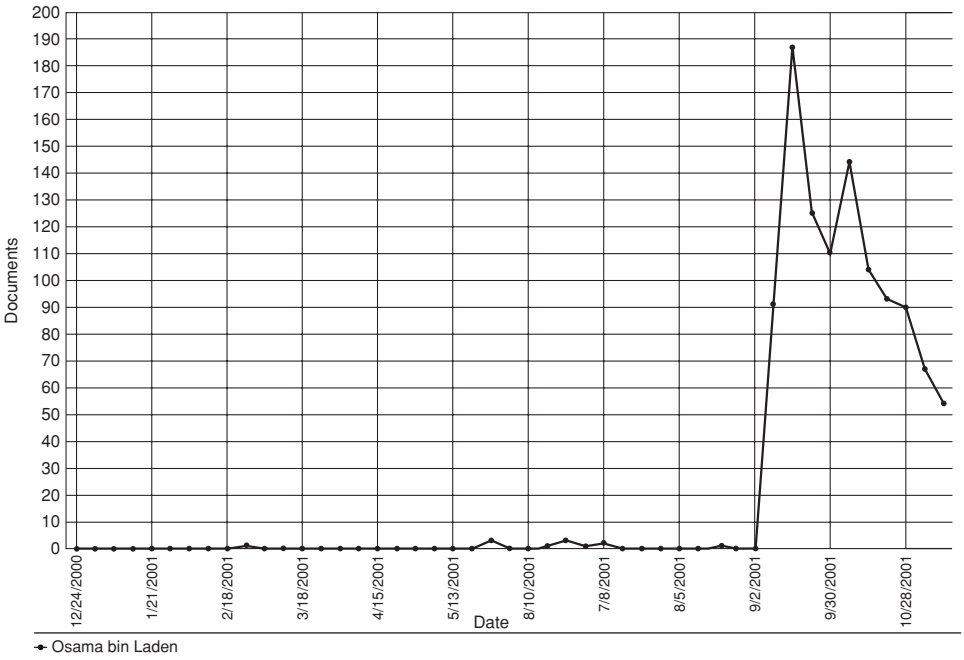

Figure X.14. Line graph showing number of documents containing the entity *Osama bin Laden* over time.

A circle graph is especially useful in visualizing patterns of association rules, though it is also very adaptable to displaying category information (Aumann, Feldman, et al. 1999). The format has been popularized by the widely used commercial data mining visualization tool *NetMap* (Duffet and Vernik 1997). Figure X.16 shows a basic circle graph.

Essentially, a circle graph is, as the name suggests, a circular graph around the circumference of which are mapped items. Relations between these items are represented by edges that connect the items across the interior area of the circle.

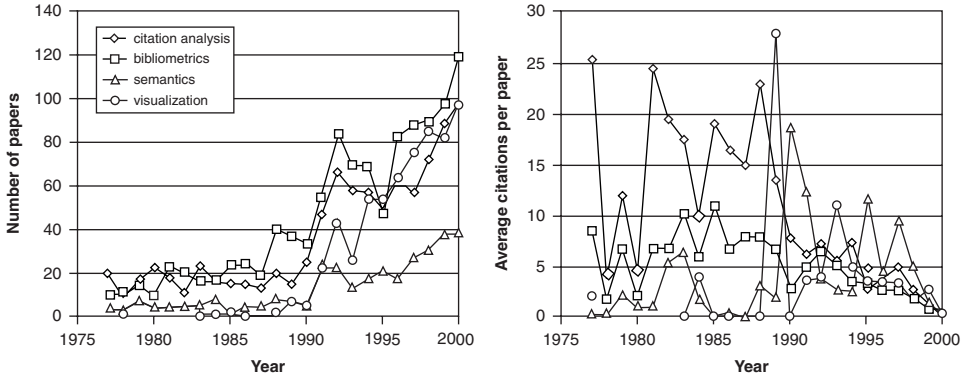

Figure X.15. Two examples of multiline graphs comparing trend lines of quantities over time. (From Borner et al. 2003. Reprinted with permission.)

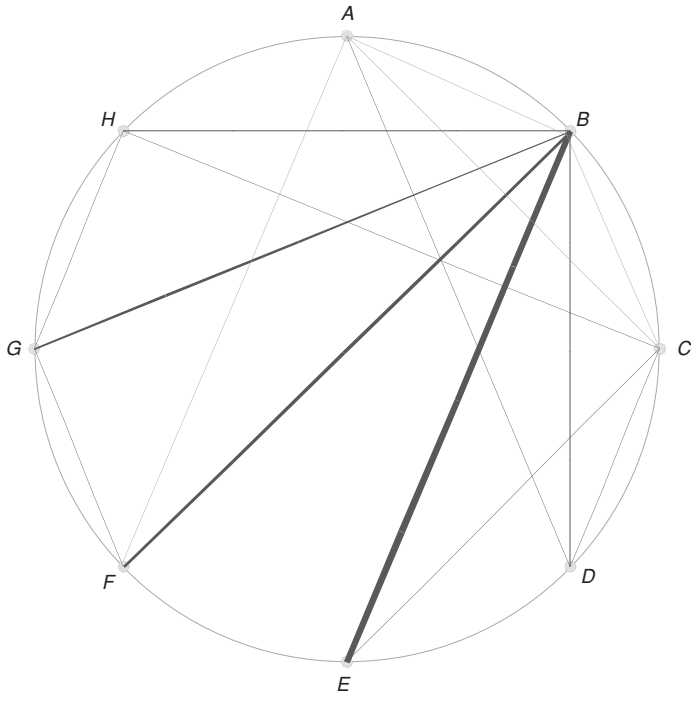

Figure X.16. Circle graph.

Style elements, such as the color and thickness of the connecting lines, can be used to correspond to particular types of information about the connection. In addition, color gradients in the connecting lines can be used to show the direction of a relationship.

Circle graphs excel at modeling association rules that appear in the answer sets to queries (see Figure X.17). It is common for individual concepts to appear as points around the rim of the circle in association-oriented circle graphs. And their association with another concept is demonstrated by a connecting edge.

Several additional visual enhancements are common in association-oriented circle graphs to enable users to have a richer graphic model of underlying textual data. First, it is common for connecting lines to use color gradients (e.g., going from yellow to blue) to show the directionality of an association. Second, a single distinct color (e.g., bright red) might also be used for a connecting line to denote a bidirectional association.

Third, the relative thickness of connecting edges may be used to suggest some corresponding information about values relating to the association. Finally, the size, color, and font type chosen for the depiction of concept names around the circumference of the circle graph can be used to communicate information visually about particular concepts in a query result set.

One method that has been used for enhancing the interactivity of an association-oriented circle graph is to make the graph's peripheral concept names and interior connecting lines "click-sensitive" jumping points to other information. A user could click or mouse-over a concept name and obtain additional ontological information

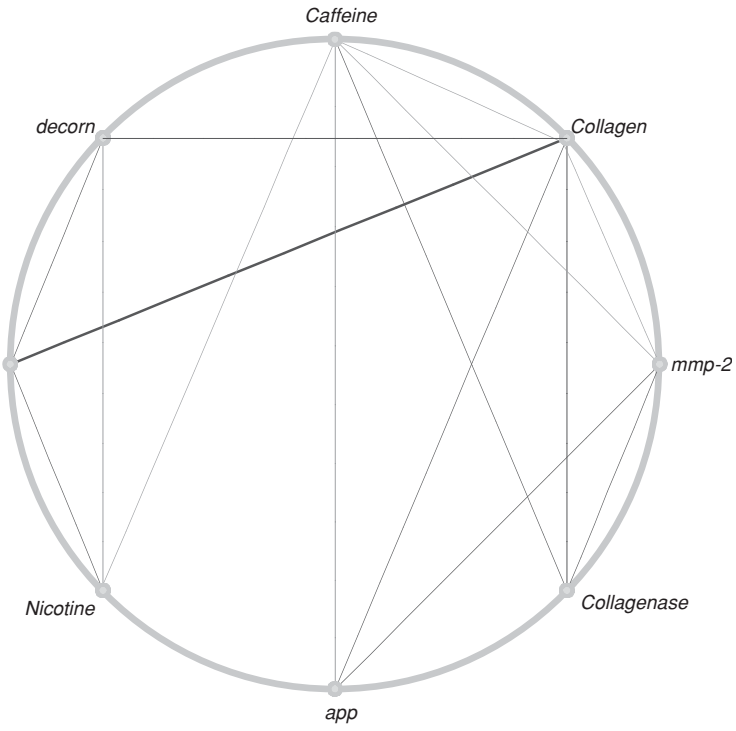

Figure X.17. Association-oriented circle graph.

relating to the concept, or, by clicking on a linking edge, a user could see the highlighted association's position in a list of associations ranked according to some quality measure.

Although circle graphs are particularly well-suited to modeling large volumes of association data, it is important to recognize that this visualization approach – like most others – can still have its effectiveness impaired by too many concepts or associations. Therefore, with circle graphs, it is often advisable to offer users easy access to controls over refinement constraints. This allows users to calibrate a circle graph's own visual feature dimensionality quickly to a level most suitable to a given search task and a particular user's subjective ability to process the visual information in the graph.

Category-Connecting Maps

Circle graphs often serve as the basis for *category-connecting maps*, another visualization tool useful in text mining. Beyond the basic taxonomical view afforded by a more traditional information retrieval-type category map, a category-connecting map builds on an association-oriented circle graph by including the additional dimension of category context for the concepts in the graph.

Category-connecting maps generally show associations between concepts in several categories – all within a particular context. For instance, Figure X.18 shows a category-connecting map with associations between individuals in the category *People* and entities in the category *Organization* – all within the context of *Terrorism*.

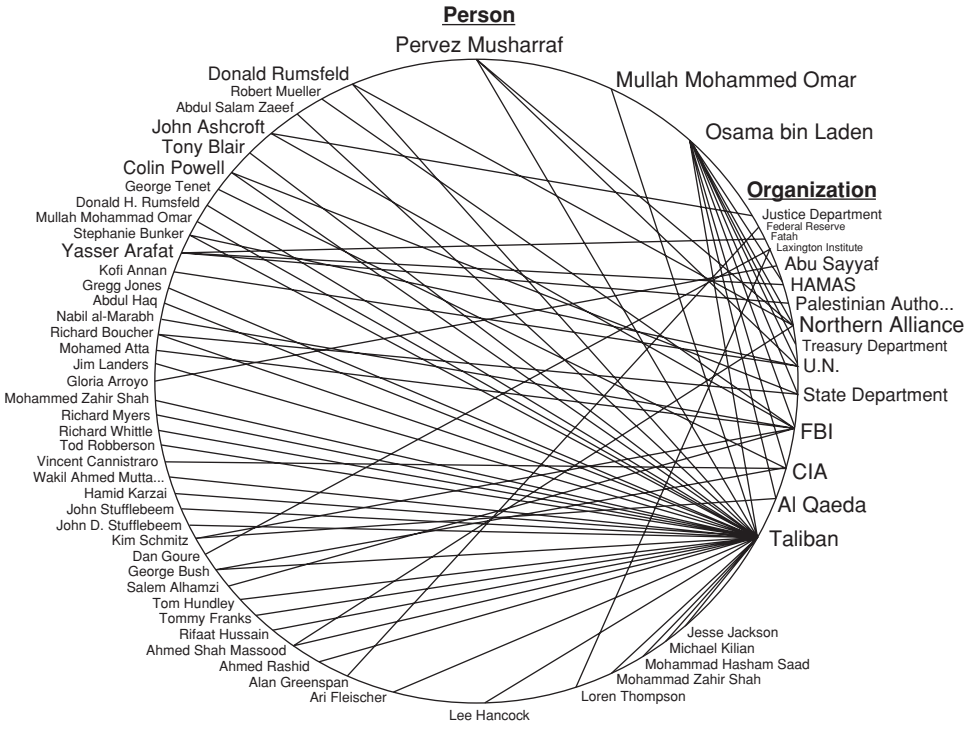

Figure X.18. Category-connecting map of associations in the context of person and organization.

In creating category-connecting maps, special attention is typically paid to the formatting of graphical and text elements on the periphery of the circle. In Figure X.18, concepts are plotted around the periphery of the circle in a way that concentrates concepts within each category together. Such concentration of concepts on the periphery of the circle can leverage preprocessed hierarchical ordering of concepts within broader "contexts" to speed rendering. However, context concepts and categories for category-connecting maps can also be generated on the fly by various algorithmic techniques ranging from the leveraging of association rules to more specialized approaches like those discussed in Chapter II relating to the generation of context graphs.

Concepts within a given category will typically have their concept labeling all formatted in the same color or font type to reinforce the visual layout technique of showing concepts within categories, and this color or font type will contrast with those used for other categories displayed in the graph. Finally, higher level category labels are often displayed to the center and outside of the "cluster" of their concepts (e.g., the category names *Person* and *Organization* are underlined and offset from the circle in Figure X.18).

Multiple Circle Graph and Combination Graph Approaches

Often, text mining applications that employ circle graphs have graphical interfaces supporting the generation and display of more than one complete circle graph at

a time. One reason for this is that, because of a circle graph's strength in showing a large amount of data about a given query set at a glance, multiple circle graphs displayed together can have tremendous value in helping establish explicit or implicit comparisons between different query results.

This advantage might be leveraged in a text mining set through the plotting of two or more circle graphs on screen at the same time, each having different refinement constraint values. Another example of this approach is category-connecting maps run against the same document collection and same main category groupings but with different contexts. Each of these examples would allow a user to make side-by-side assessments of differences and similarities in the graphing patterns of multiple graphs. Figure X.19 illustrates the use of multiple circle graphs in a single visualization.

Another technique that relies on showing multiple circle graphs on screen at the same time results from trying to emphasize or isolate "subgraphs," or to do both, from within a circle graph. For instance, because circle graphs can be used to model so much data all at once, some more subtle relationships can become de-emphasized and obscured by the general clutter of the graph. By allowing a user to click on several items that are part of a main circle graph, a text mining system can offer subgraphs that display only the relationships between these items. Being able to see such subgraphs discretely while viewing the main circle graph as a whole can lead to new forms and levels of user interactivity.

Similarly, circle graphs can benefit from side-by-side comparisons with other graphs. For instance, instead of limiting a text mining system's graphing options to circle graphs and their subgraphs, one could also use simple concept association graphs to graph highlighted relationships shown within an association-oriented circle graph or concept-connecting map.

X.3.6 Self-Organizing Map (SOM) Approaches

Text mining visualization has benefited from contributions made by research into how artificial neural networks can aid information visualization. Perhaps one of the most important of these contributions is the paradigm of self-organizing maps, or SOMS, introduced by Kohonen in 1982 and first applied in 1991 to problems in information visualization in Lin, Soergel, and Marchionini (1991).

SOMs are generated by algorithms that, during a learning phase, attempt to iteratively adjust weighting vectors derived from the relationships found in a high-dimensional statistical data input file into various forms of two-dimensional output maps. Because of this approach, SOMs have advantages in treating and organizing data sets that are extremely large in volume and connecting relationships.

WEBSOM

One of the most widely known and used applications of SOMs to textual data is WEBSOM. WEBSOM uses an adapted version of Kohonen's original SOM algorithm to organize large amounts of data into visualizations that applications designers refer to as "document maps," which are essentially graphical models similar to topographical maps (see Figure X.20).

Shading on the map face displays concentrations of textual data around or near a particular keyword or concept; lighter areas show less concentration. Hence, the

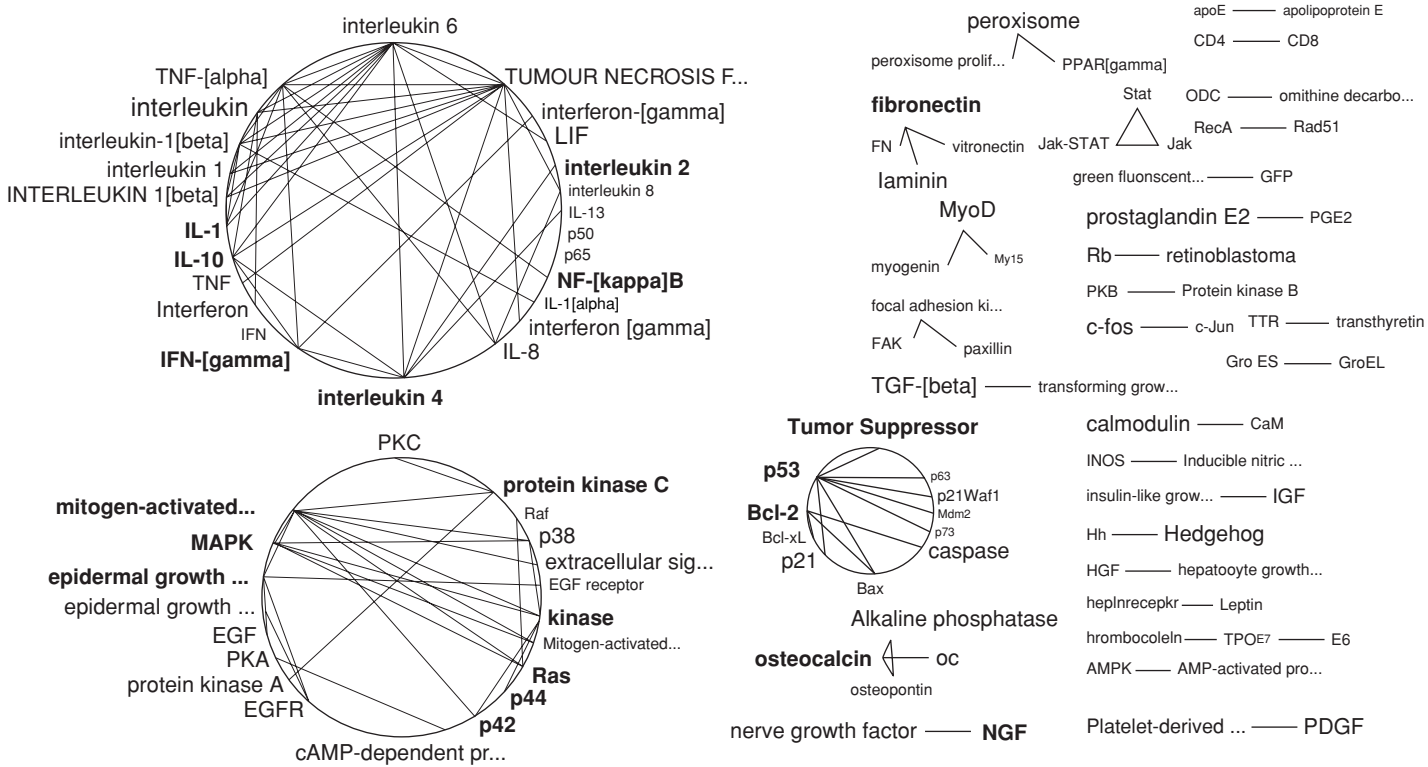

Figure X.19. Side-by-side circle graphs with subgraphs. (From Feldman, Regev, et al. 2003.)

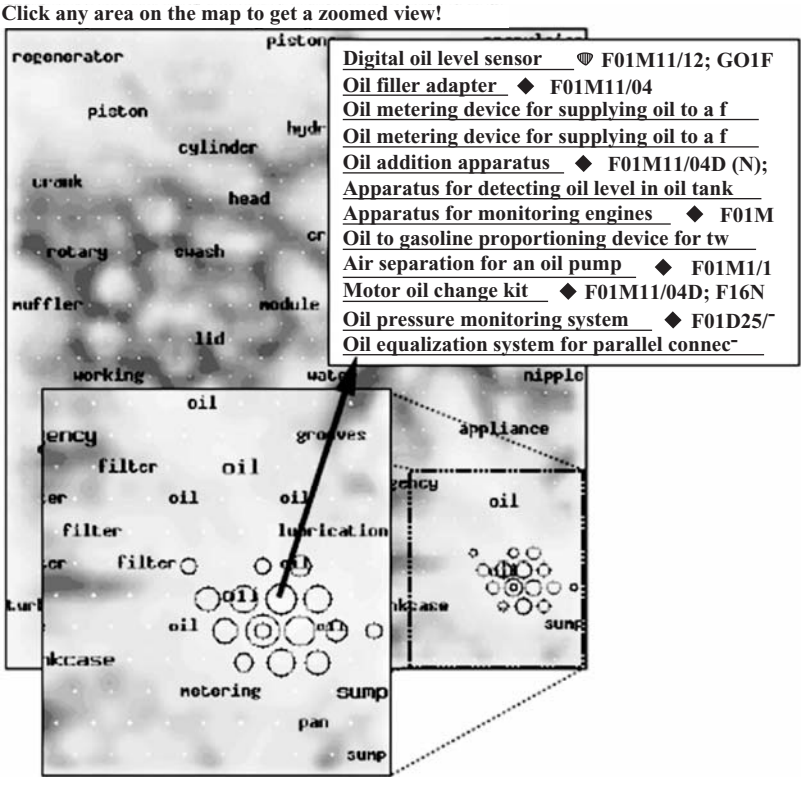

Figure X.20. WEBSOM – during queries, users are guided by a document map via visual cues in a GUI that supports both interactive zooming and browsing support functions. (From Lagus et al. 1999. Reprinted with permission of Springer Science and Business Media.)

graphical approach represented by WEBSOM is particularly suited to text mining analytics involving some element of reference to a category. However, the system has proven flexible enough to be applied to other tasks as well. Moreover, although WEBSOM may initially have been geared more toward solving problems in information retrieval for high-dimensionality document collections, academic attention has been devoted specifically to its uses as a toolkit for building text mining interfaces. A basic WEBSOM document map can be seen in Figure X.21.

One of the strongest advantages of WEBSOM – and similar SOM-based systems that it has inspired – has been a proven ability to handle large amounts of data. WEBSOM has built a document map to address more than one million documents, and its automatic algorithm for building document maps is reportedly able to complete a visualization for a small-to-modest document collection (approximately 10,000 documents) in less than a few hours.

Another advantage of WEBSOM is the robustness of the interface's functionality. WEBSOM is a fully zoomable interface that enables sections of a full document map to be repeatedly zoomed at various levels of magnification. The document map GUI is also very interactive in that clicking on a highlighted concept identifier or a section of the document map will bring up lists of corresponding documents, statistical information about the documents, or both typically in a pop-up or separate window.

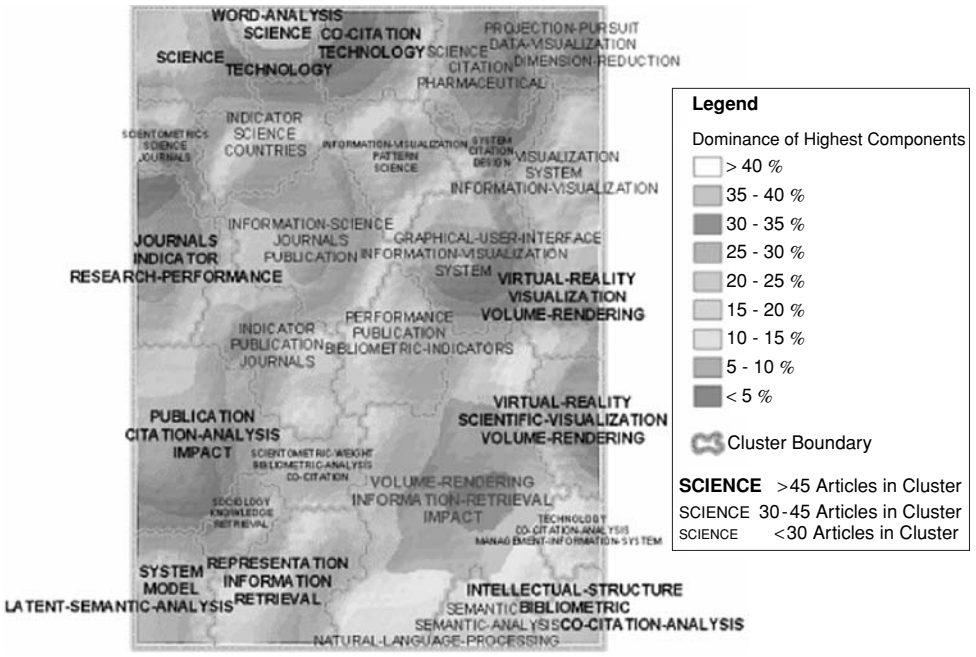

Figure X.21. WebSOM-like cartographic document map with typical graph legend. (From Borner et al. 2003. Reprinted with permission.)

SOM Algorithm

Honkela (1997) has summarized the SOM algorithm along the following lines:

- Assume an input dataset of concepts is configured as a table, with the intended output being the mapping of these data onto an array of nodes. The set of input data is described by a vector $X(t) \in R^n$, where t is the index of the input data. In terms of output, each node i in the map contains a model vector $m_i(t) \in R^n$; this model vector has the same number of elements as the input vector $X(t)$.
- The SOM algorithm is stochastic and performs a regression process. Therefore, the initial values of the elements of the model vector, $m_i(t)$, may be selected at random.
- Input data are mapped into a location in the output array, the $m_i(t)$ of which "matches" best with $x(t)$ in some metric. The SOM algorithm creates an ordered mapping by repeating the following tasks:
  - An input vector $x(t)$ is compared with all the model vectors $m_i(t)$; the best-matching element (node) on the map (i.e., the node where the model vector is most similar to the input vector according to some metric) is discerned. The best-matching node on the output map is sometimes referred to as the "winner."
  - The model vectors of the winner and a number of its neighboring nodes (sometimes called "neighbors") in the array are changed toward the input vector according to a customized learning process in which, for each data element input vector $x(t)$, the winner and its neighbors are changed closer to $x(t)$ in the input data space. During the learning process, individual changes may

actually be contradictory, but the overall outcome in the process results in having ordered values for $m_i(t)$ gradually appear across the array.

- Adaptation of the model vectors in the learning process takes place according to the following equations:

$$m_i(t+1) = m_i(t) + \alpha(t)[(t) - m_i(t)] \text{ for each } i \in N_c(t);$$
$$\text{otherwise, } m_i(t+1) = m_i(t),$$

where t is the discrete-time index of the variables, the factor $\alpha(t) \in [0, 1]$ is a scalar that defines the relative size of the learning step, and $N_c(t)$ describes the neighborhood around the winner node in the map array. Typically, at the beginning of the learning process, the radius of the neighborhood can be quite large, but it is made to consolidate during learning.

One suggested method for examining the quality of the output map that results from the running the SOM algorithm is to calculate the average quantization error over the input data, which is defined as $E\{\|X - m_c(X)\|\}$, where c indicates the best-matching unit (sometimes referred to as the BMU) for x. After training, for each input sample vector, the BMU in the map is searched for, and the average of the individual quantization errors is returned.

Several deficiencies, however, have been identified in WEBSOM's approach. Some have pointed out that WEBSOM's algorithm lacks both a cost function and any sophisticated neighborhood parameters to ensure consistent ordering. From a practical standpoint some have commented that a user can get "lost" in the interface and its many zoomable layers. In addition, the generalized metaphor of the topographical map is not a precise enough aid in displaying patterns to support all text mining pattern-identification functions.

X.3.7 Hyperbolic Trees

Initially developed at the Xerox Palo Alto Research Center (PARC), *hyperbolic trees* were among the first *focus and context* approaches introduced to facilitate visualization of large amounts of data. Relying on a creative interpretation of Poincaré's model of the hyperbolic non-Euclidean plane, the approach gives more display area to a part of a hierarchy (the focus area) while still situating it within the entire – though visually somewhat de-emphasized – context of the hierarchy. A widely known commercial toolkit for building hyperbolic tree visualization interfaces is marketed by Inxight Software under the name *StarTree Studio* (see Figure X.22).

Hyperbolic tree visualizations excel at analysis tasks in which it is useful for an analyst to see both detail and context at the same time. This is especially true in situations in which an analyst needs to traverse very complex hierarchically arranged data or hierarchies that have very large amounts of nodes.

Other more common methods for navigating a large hierarchy of information include viewing one page or "screen" of data at a time, zooming, or panning. However all of these methods can be disorienting and even distracting during intensive visual data analysis. A hyperbolic tree visualization allows an analyst always to keep perspective on the many attached relationships of a highlighted feature.

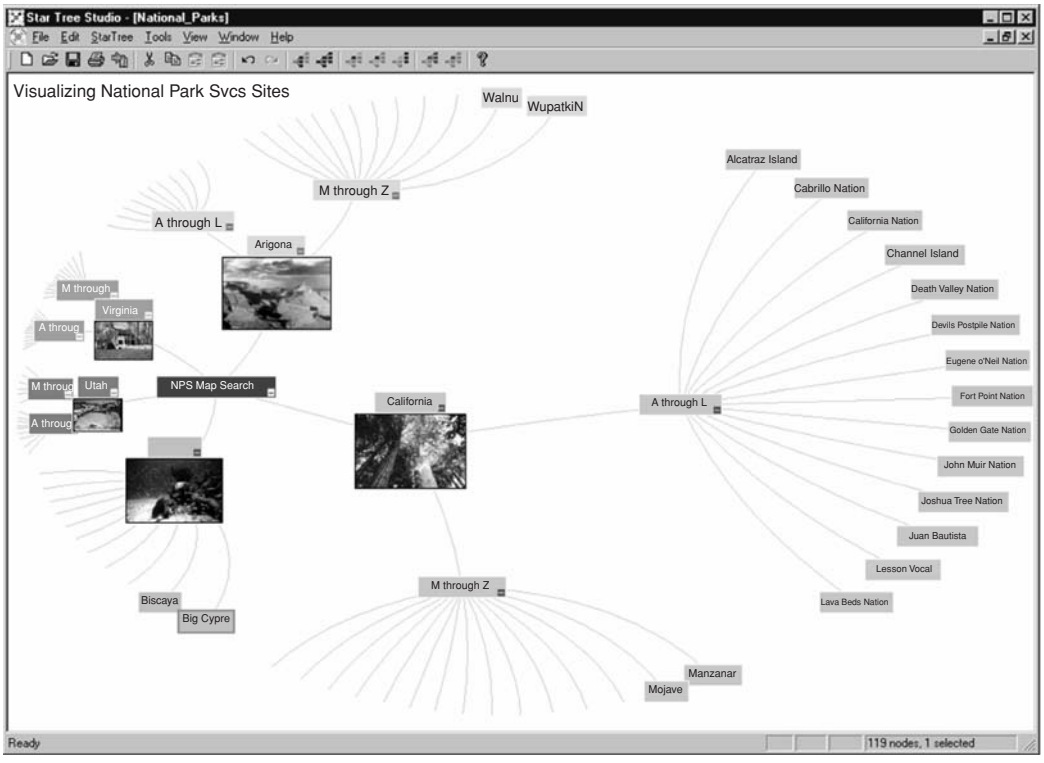

Figure X.22. Hyperbolic tree for visualizing National Park Service sites. (From Inxight StarTree Studio. Copyright Inxight Software.)

Hyperbolic tree visualization tools have from their inception been designed to be highly dynamic aids for textual data exploration. Initially, a hyperbolic tree diagram displays a tree with its root at the center of a circular space. The diagram can be smoothly manipulated to bring other nodes into focus. The main properties that support the capabilities of the hyperbolic tree are that

- elements of the diagram diminish in size as they move outward, and
- there is an exponential growth in the number of potential components.

Effectively, these two properties might be described as a form of fisheye distortion and the ability to uniformly embed an exponentially growing structure. Together, they allow the hyperbolic tree visualization tool to leverage Poincaré mapping of the non-Euclidean plane to explore very large hierarchies in a visualization frame relatively limited in size.

The hyperbolic tree's peculiar functionality does more than simply allow a user to interact with a larger number of hierarchical nodes than other more traditional methods or to view a highlighted feature with reference to a richer amount of its context. It also very much encourages hands-on interaction from a user with a hierarchical dataset. Figure X.23 shows another example of a hyperbolic tree.

By enabling a user to stretch and pull a complex hierarchy around a focused-on item with a mouse and then skip to another node with a mouse click and view the rest of its hierarchy context at various angles, a hyperbolic tree diagram promotes dynamic, visualization-based browsing of underlying data. Instead of being

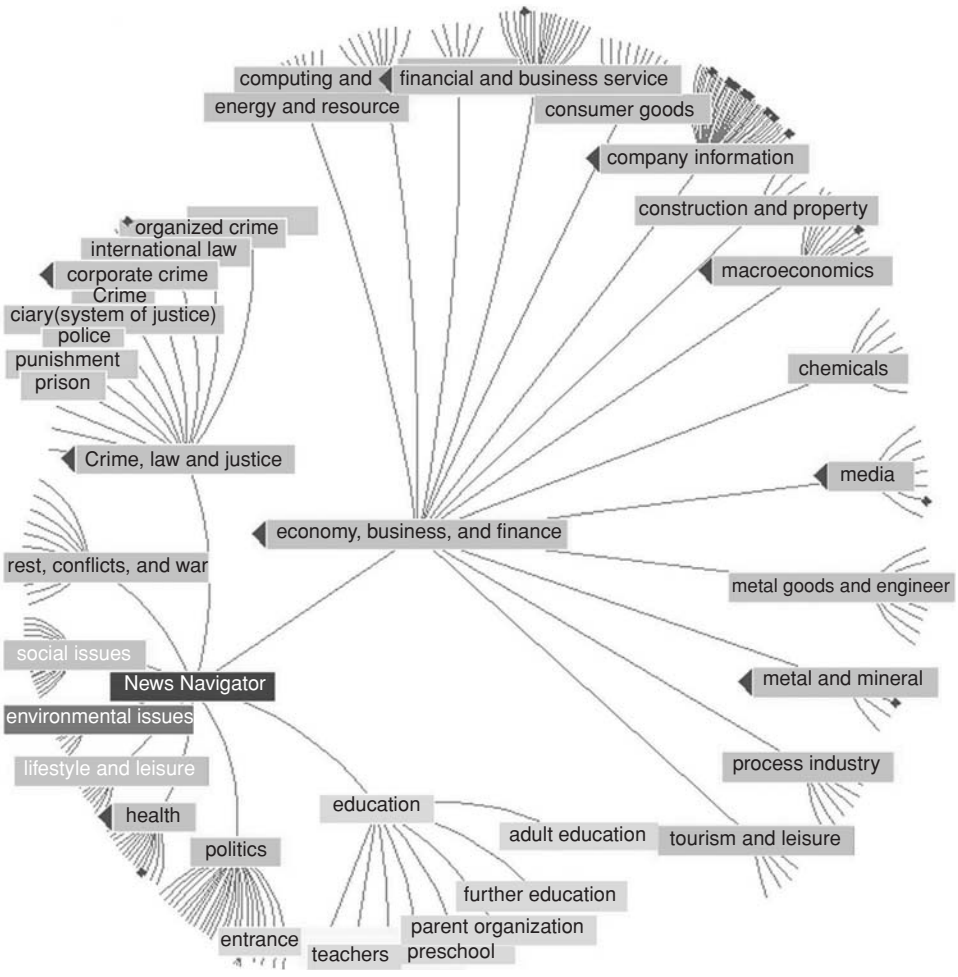

Figure X.23. Hyperbolic tree visualization of a document collection composed of news articles. (Courtesy of Inxight Software.)

semipassive viewers and inputters of query data, users become fully engaged participants in pattern exploration.

X.3.8 Three-Dimensional (3-D) Effects

Many text mining systems have attempted to leverage three-dimensional (3-D) effects in creating more effective or flexible visualization models of the complex relationships that exist within the textual data of a document collection. 3-D visualizations offer the hope that, by increasing the apparent spatial dimensionality available for creating graphic models of representations such as those produced by more complex, second-generation, multiple-lattice SOMs, users may be able to examine and interact with models that make fewer compromises than are required by traditional (2-D) hierarchical or node-and-edge representations.

Moreover, higher powered 3-D rendering algorithms and wider availability of sophisticated workstation graphics cards create the conditions for making such 3-D visualizations more practical for use in text mining systems than ever before. In

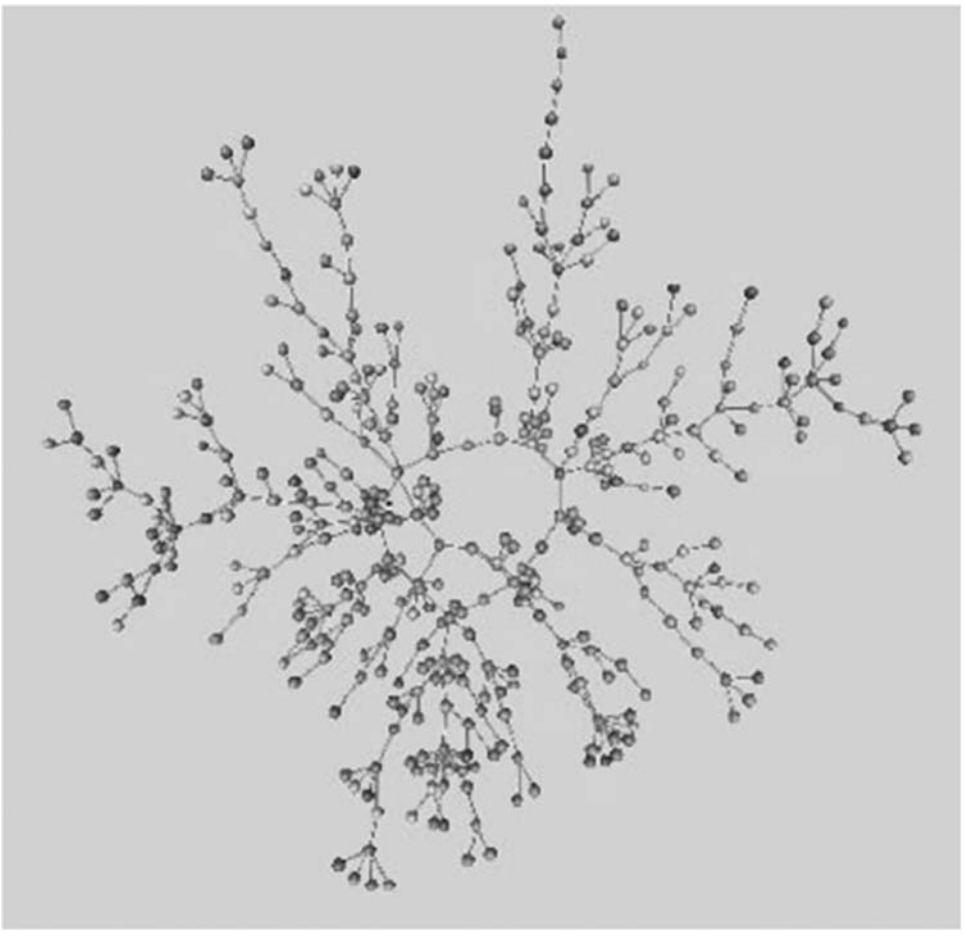

Figure X.24. A 3-D overview of a scientific author cocitation map suggesting a diverse, unconcentrated domain. (From Borner et al. 2003.)

addition, to supplement navigation in a non-2-D environment, many relatively lightweight VR rendering and exploration environments are available now that can easily be embedded into front-end interfaces. An example of a 3-D network map can be seen in Figure X.24.

Despite all of the potential opportunities offered by 3-D treatments of models for information visualization, these models also introduce several new challenges and problems. Two significant problems for using 3-D visualization approaches in text mining are occlusion and effective depth cueing (see example in Figure X.25). Both of these problems are exacerbated in situations in which presentation of high-dimensional data is required. Unfortunately, such situations are precisely those in which text mining applications will be likely to incorporate 3-D visualization tools.

Moreover, 3-D visualizations do not generally simplify the process of navigating and exploring textual data presentations. Often, some level of specialized navigational operations must be learned by the user of a 3-D projection or VR-based visualization. This can become something of a barrier to inspiring intuitional

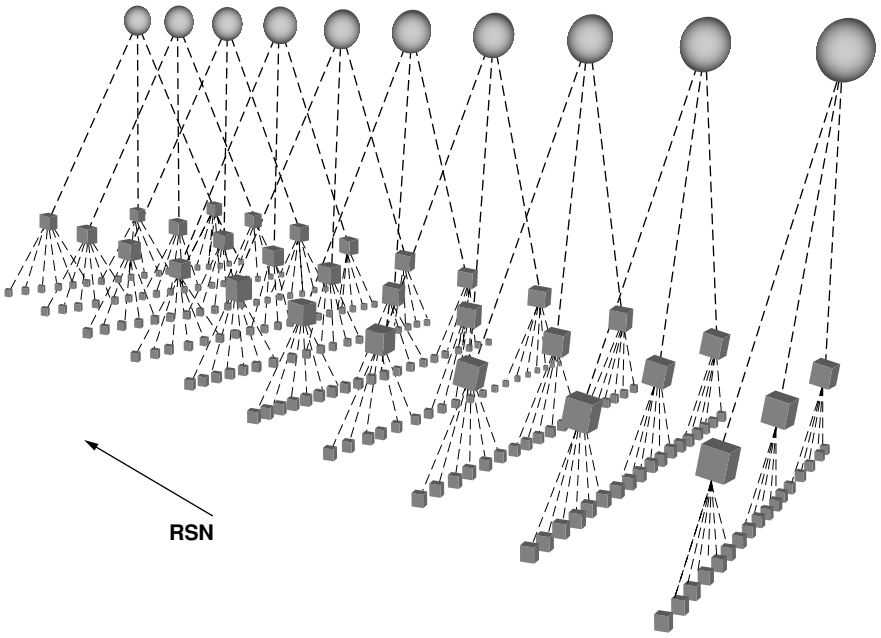

Figure X.25. Presentation from a visualization suggested by Gall et al. (1999). Visualization represents hierarchies over time. However, challenges of element occlusion occur at lower levels of the various hierarchies. Such a presentation may also not be understood intuitively by users. (From Graham 2001. Reprinted with permission, © 2001 IEEE.)

navigation of a data presentation. Increased complexity in navigation is generally inversely proportional to stimulating high levels of iterative user interaction.

There is no doubt that the verdict will be out for some time on the impact of 3-D treatments on text mining visualization. Certainly, there will be a great deal of continued research and experimentation in an effort to make 3-D approaches more practically useful. Currently, the disadvantages of 3-D approaches outweigh the proposed advantages; 3-D visualization may never be very useful in comprehending nonspatial information like that encountered in text mining.

In the short term, designers of text mining systems should carefully evaluate the practical benefits and drawbacks inherent in the potential inclusion of a 3-D visualization tool in their applications before being carried away by the theoretical advantages of such new graphical tools.

X.3.9 Hybrid Tools

In the discussion of circle graphs in Section X.3.5, it was noted that sometimes combinations of identical multiple graphs or different types of multiple graphs have a special role in providing analytical information about the results from a text mining query's answer set. Designers of visualization tools often come up with presentation techniques that might be seen as hybrid forms incorporating components of different visualization formats into a coherent, new form. Three creative examples of hybrid visualization approaches can be seen in Figures X.26, X.27, and X.28.

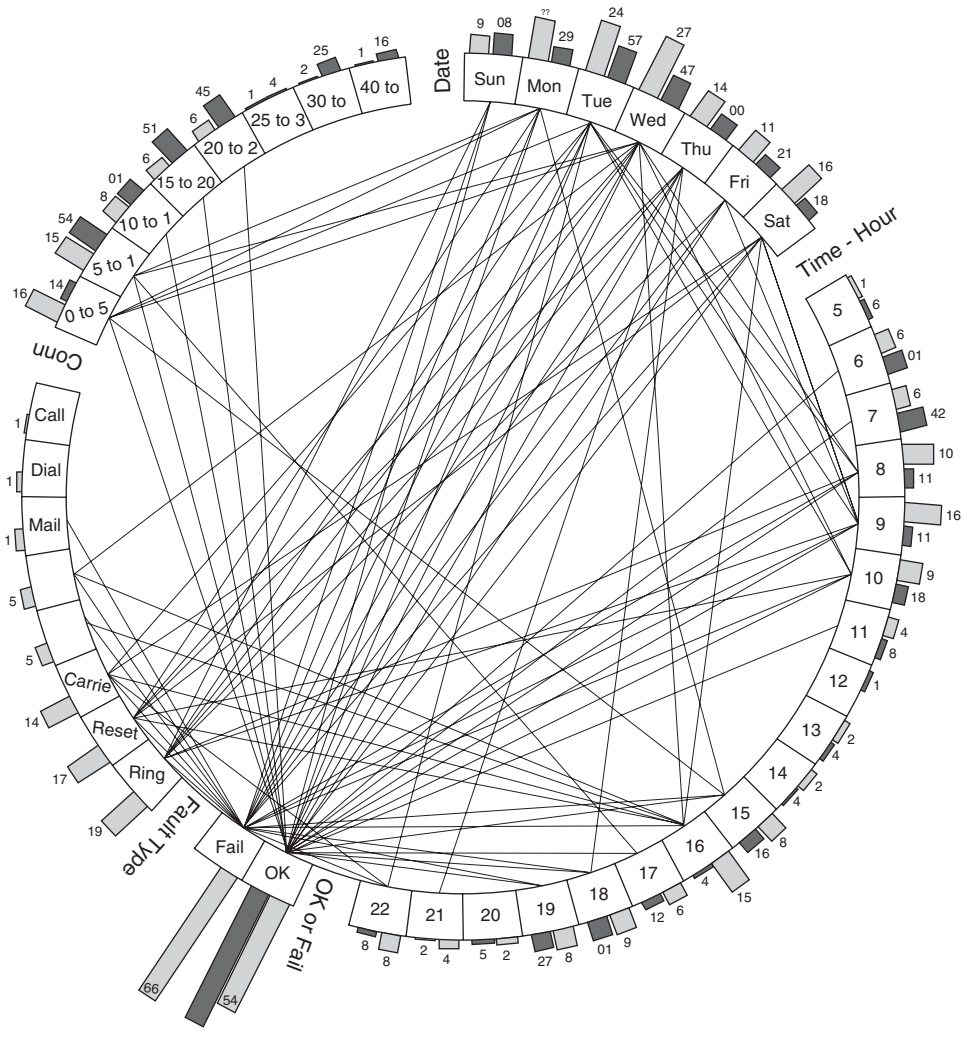

Figure X.26. Daisy chart combining aspects of a circle graph and complex comparative histogram. (Reprinted with permission from James Miller.)

One critical driver for the innovation of such forms is the desire to achieve more presentation concision. By supplementing currently well-known graphical formats with additional new elements, one might at least theoretically be able to increase dramatically the amount of information communicated by the graphical tool.

One of the possible pitfalls in creating such hybrid forms is overcomplication; ideally, users should be able to understand the major "messages" communicated by a presentation approach without too much potential for confusion. Another potential pitfall is decreased visual clarity of a graph; because text mining visualizations so often involve overabundance in patterns, more complex visualizations can also result in greater presentation "clutter" issues.

Because most designers of text mining systems actually implement visualization approaches initially developed by information visualization specialists, these considerations should be weighed when evaluating the possible graphical tool alternatives

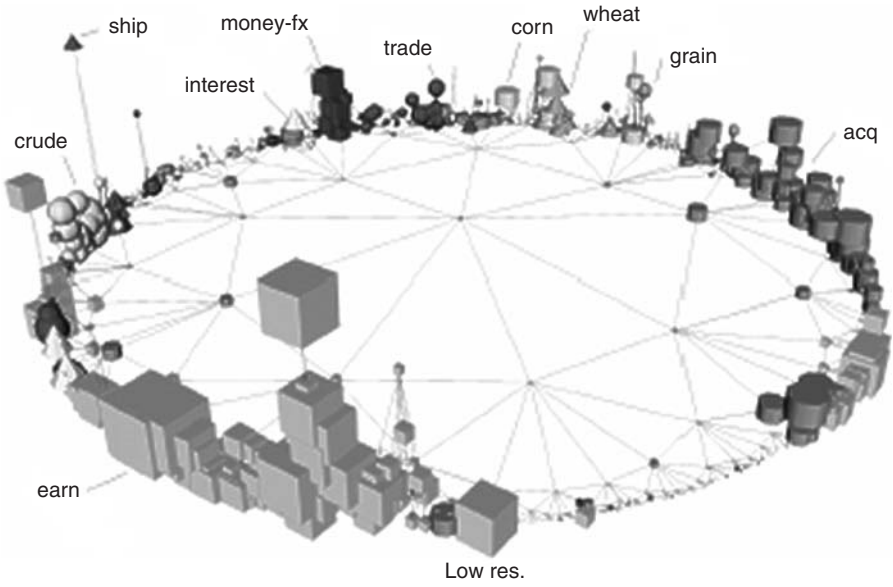

Figure X.27. View of an HSOM or hyperbolic self-organizaing map that projects 3-D elements on a triangularly tesselated hyperbolic tree grid. (From Ontrup and Ritter 2001a. Reprinted with permission of The MIT Press.)

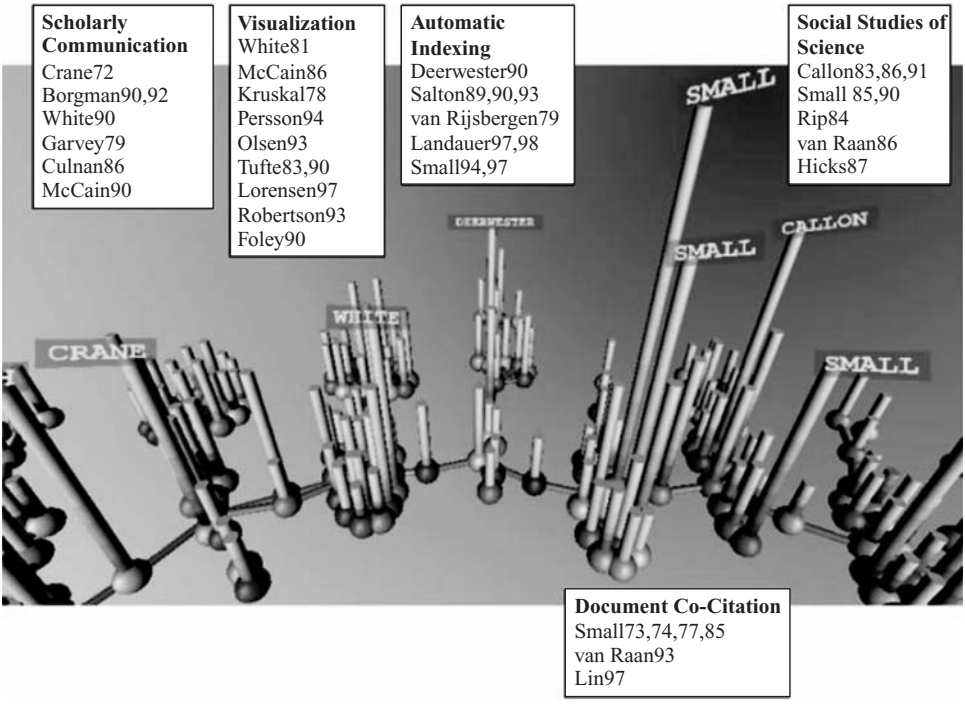

Figure X.28. Specialized network diagram that includes elements of nodes and links graphs and histogrammatic presentation with 3-D effects and character-based tables. From the Arist Co-Citation Project. (From Borner et al. 2003. Reprinted with permission.)

for a given text mining application. Designers of text mining systems need to be able to consider several commercial or freeware visualization alternatives from the perspective of their text mining system's intended functionality.

Above all, these system designers need to be careful to avoid the temptation of having "a visualization solution in search of a text mining problem." Creating the conditions for maximum interaction from a user depends on ensuring a more seamless, compatible fit between a visualization approach's strengths and the algorithmic search techniques that form the core of the text mining situation a particular presentation layer is meant to support.

For this reason, it sometimes does pay to look for hybrid approaches. A special-purpose hybrid visualization form may meet the needs of a very specific text mining application in ways better than more generic forms.

X.3.10 Citations and Notes

Sections X.3.1–X.3.3

The Document Explorer application is described in Feldman, Kloesgen, and Zilberstein (1997a) and Feldman, Fresko, Hirsh, et al. (1998) and summarized in Section VI.5.1.

Discussions of relevant hierarchical visualization approaches can be found in Karrer and Scacchi (1990); Johnson and Shneiderman (1991); Robertson, Mackinlay, and Card (1991); and Hearst and Karadi (1997). *Simple concept graphs* are an updating of the simple keyword graphs introduced in Feldman, Kloesgen, and Zilberstein (1997a).

A good general discussion of some of the considerations employing DAGs in information visualization can be found in Melancon and Herman (2000), in which the authors make several useful points, including the following: (a) DAGs might be seen as a natural generalization of tree structures, (b) aesthetically pleasing drawings of DAGs are those with the minimum possible number of edge crossings (though this can sometimes be difficult to manage in graphing large datasets), and (c) DAGs can serve as a a kind of intermediary form between tree structures and general graphs. For a review of a DAG-generating program that creates the type of DAG visualizations described and illustrated in Section X.3.2, see Gansner, North, and Vo (1988). See also Gansner et al. (1993).

All references to Zhou and Cui's DAG-based representations of elements of data from the GO Consortium's Gene Ontology are from Zhou and Cui (2004). Information on the Gene Ontology can be found in GO Consortium (2001).

Sections X.3.4–X.3.7

The two examples of the multiline graphs shown in Figure X.15 are directly from Borner, Chen, and Boyack (2003). At least one examination of 2-D histograms in text mining has suggested that they are not especially useful at displaying some basic types of query results relating to association rules (Wong et al. 2000).

Aumann et al. (1999) provides an early treatment of circle graphs in text-oriented knowledge discovery. Rainsford and Roddick (2000) underscores the comprehensive "at-a-glance" property that circle graphs have in concisely showing an entire representation of relationships in large amounts of data. The NetMap circle graph

information visualization tool is described in Duffet and Vernik (1997). Information about commercial Netmap products is available at <www.netmap.com>.

Important background reference materials on SOMs include Kohonen (1981), Kohonen (1982), Lin et al. (1991), Lin (1992), Kohonen (1995), Kohonen (1997), Lin (1997), Kohonen (1998), and Lagus (2000b). Background reference materials on WEBSOM include Honkela et al. (1997); Honkela, Lagus, and Kaski (1998); Lagus (1998); and Lagus et al. (1999). The SOM algorithm described in Section VI.3.6. has been summarized from Honkela (1997).

Beyond WEBSOM, many systems and computer science research projects have incorporated SOM-style visualizations. Some representatives of the wide influence of SOM-style interfaces can be seen in Merkl (1998), Borner et al. (2003), and Yang, Chen, and Hong (2003).

Hyperbolic trees are introduced and discussed in Lamping and Rao (1994); Lamping, Rao, and Pirolli (1995); and Munzner and Burchard (1995). StarTree Studio is a product of Inxight Software; additional product information can be found at <www.inxight.com>. All images from StarTree Studio are the property of Inxight Software. The hyperbolic tree representation of the Internet comes from Munzner and Burchard (1995). Another interactive "focus + context" approach, the Table Lens, is discussed in Rao and Card (1994).

Section X.3.8–X.3.9

Borner et al. (2003) provides a brief but practical review of some 3-D approaches used in visualizing knowledge domains that would also be applicable to text mining activities Koike (1993) is another useful source. The effects of such things as potential drawbacks as occlusion and effective depth cueing in 3-D visualizations are discussed in Rokita (1996) and Hubona, Shirah, and Fout (1997).

The visualization in Figure X.25 appears as a reference in Graham (2001) that originally appeared in Gall et al. (1999). Graham (2001) points out that there is growing consensus that 3-D visualizations are not that useful in comprehending non-spatial information, whereas Cockburn (2004) seems to suggest the opposite view.

The Daisy Chart displayed in Figure X.26 is a visualization copyrighted by James Miller of Daisy Analysis (<www.daisy.co.uk>). The daisy chart also appears in Westphal and Bergeron (1998).

Figure X.27 illustrates one application of the hyperbolic self-organizing map or HSOM. The HSOM is discussed in Ontrup and Ritter (2001a, 2000b).

The hybrid 3-D network diagram illustrated in Figure X.28 comes from Borner et al. (2003).

X.4 VISUALIZATION TECHNIQUES IN LINK ANALYSIS

Although the discipline of link analysis encompasses many activities, several specific tasks are frequently addressed by a few specialized visualization approaches. In particular, these tasks include

- analysis of a single known concept for the relatedness to, or degrees of separation from, other concepts, and
- the identification and exploration of networks or pathways that link two (or more) concepts.

Although various generic text mining activities generally involve, as a primary exploratory approach, the investigation of query result sets in a browser supplemented by visualization techniques, current, state-of-the-art link analysis methods almost always depend on the visualization approach as a central operation. The exploration of pathways and patterns of connectedness is substantially enhanced by visualizations that allow tracking of complex concept relationships within large networks of concepts.

Chapter XI focuses on essential link analysis concepts such as paths, cycles, and types of centrality and also offers a detailed, running example involving the construction of a model of a social network appropriate to link analysis activities in the form of a spring graph. Although spring graphs are certainly one of the more common graphing forms used in link analysis, many visualization techniques have been applied in this quickly evolving discipline. This section surveys some visualization approaches that have been adapted to support link analysis.

X.4.1 Practical Approaches Using Generic Visualization Tools

Developers of graphical interfaces to aid in link detection and analysis often slightly modify more generic visualization formats to orient these graphic approaches more toward link detection activities. In particular, simple concept graphs, circle graphs, and hyperbolic trees have been applied to and, in some cases, modified for the support of link detection tasks. Even histograms and line graphs have been put into service for link analytics.

For example, a common simple concept association graph could be used to show *persons* associated with *organizations* within the context of some other concept. Such a graph could be oriented toward link detection activities by centering the graph on a single known person and allowing the outwardly radiating edges and vertices to constitute a relationship map.

In a sense, this type of manipulation of the simple concept association graph creates at least an informal *focus* for the graph. Ease of following the relationships in the map can be enhanced by stylistic devices: person nodes and labels and concept nodes and labels could be drawn in contrasting colors, edge thickness could be determined by the number of documents in which an association between two linked concepts occurs, and so on.

Figure X.29 shows the results of a query for all *person* concepts with associations to *organization* concepts within the context of the concept *terrorism* within a given document collection. After a simple concept association graph was generated from the results of the query, a single person concept, Osama bin Laden, was highlighted and drawn to form a central "hub" for the diagram. All person concepts were identified in a darker typeface, whereas all organization concepts were denoted by a lighter typeface.

An analyst can traverse relationships (associations) emanating from Osama bin Laden in a quick and orderly fashion. The methodology also has the advantages of being relatively quick to implement and, often, requiring only some customization of the more standard visualization approaches found bundled with most text mining-type applications.

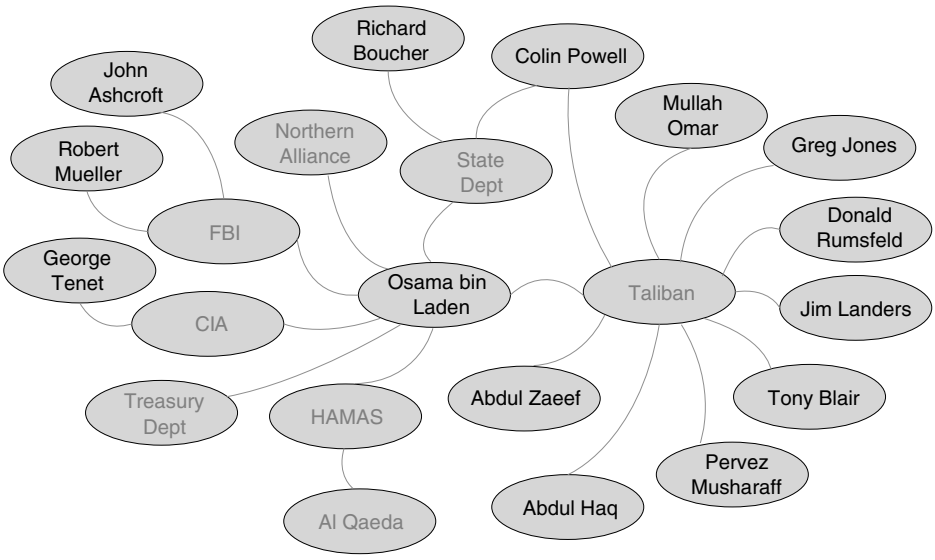

Figure X.29. Graphing results to a search query for all *Person* concepts with associations to organization concepts within the context of the concept terrorism with the concept *Osama bin Laden* as central vertex.

Of course, there are some notable limitations to this approach. First, there are a rather limited number of nodes radiating out from a central "hub" node that a user can take in at any one time. This limitation can be offset somewhat by zooming and panning capabilities.

Second, there is no sophisticated or automatic weighting methodology for emphasizing stronger or more interesting associations by some sort of visual proximity cue within a confined and manageable visualization space. This is a particularly limiting factor in the case of very large node-and-edge graphs. Certainly, one can easily increase line density between nodes or prune nodes from the graph altogether based on some quality measures.

In graphs in which very large numbers of nodes and associations are present, there is significant risk that these two limitations will prevent the user from maintaining his or her focus on the central node (because of the need to pan, page down, or zoom to a very large graph) and receiving much information about the comparative relatedness of nodes to the central focus node by means of strong spatial or proximic visual cues. Other types of specialized visualization formats do relatively better jobs in addressing these limitations.

X.4.2 "Fisheye" Diagrams

Fisheye diagrams show a distorted, lenslike view of a graph to highlight ostended "focal point" detail while maintaining relatively easy viewing of its broader, more global visual context. The term "fisheye" derives from the diagram's analogy to the super-wide-angle or fisheye lens used in photography (fisheye lenses magnify the image at the focal point while de-emphasizing, but still showing, images at the

periphery). Fisheye views of data were first proposed by Furnas in 1981 and substantially enhanced by Sarkar and Brown (1992).

The best fisheye approaches to visualizing data attempt to balance local or highlighted detail with a global context. Fisheye approaches have been described as being divisible into two categories: distorting and filtering fisheyes. Distorting fisheyes adjust the size of various graphical elements in a diagram to correspond to their interestingness, whereas filtering fisheyes de-emphasize or suppress the display of less interesting data.

Distorting Fisheye Views

Fisheye diagrams have vertices and edges, like node-and-edge graphs, but must accommodate three main ideas:

- The position of a given vertex in a fisheye view depends on its *position in the "normal view"* of the diagram and *its distance from the fisheye view's focus*.
- The size of a given vertex in the fisheye view depends on its *distance from the focus*, its *size in the normal view*, and *a value representing the relative importance of this vertex in the global structure*.
- The amount of detail in a vertex depends on its size in the fisheye view.

Sarkar and Brown (1992) formalized these concepts in the following way:

1. The position of vertex v in the fisheye view is a function of its position in normal coordinates and the position of focus f:

$$P_{\text{feye}}(v, f) = F_1(P_{\text{norm}}(v), P_{\text{norm}}(f)).$$

2. The size of the vertex in the fisheye view is a function of its size and position in normal coordinates, the position of the focus, and its *a priori importance*, or *API*, which is a measure of the relative importance of the vertex in the global structure:

$$S_{\text{feye}}(v, f) = F_2(S_{\text{norm}}(v), P_{\text{norm}}(v), P_{\text{norm}}(f), API(v)).$$

3. The amount of detail to be shown for a vertex depends on the size of a vertex in the fisheye view and the maximum detail that can be displayed:

$$DTL_{\text{feye}}(v, f) = F_3(S_{\text{feye}}(v), DTL_{\text{max}}(v)).$$

4. The visual worth of a vertex depends on the distance between the vertex and the focus in normal coordinates and on the vertex's API:

$$VW(v, f) = F_4(P_{norm}(v), P_{norm}(f), API(v)).$$

Fisheye diagrams represent a good fit with the visualization requirements of many link analysis tasks. By applying a fisheye treatment to vertices of a graph that are interesting to a user, he or she can scan, without visual interruption or panning, among many contextual relationships, as shown in the diagrammatic elements presented in the periphery of the graph. Figure X.30 shows some fisheye treatments of a SOM.

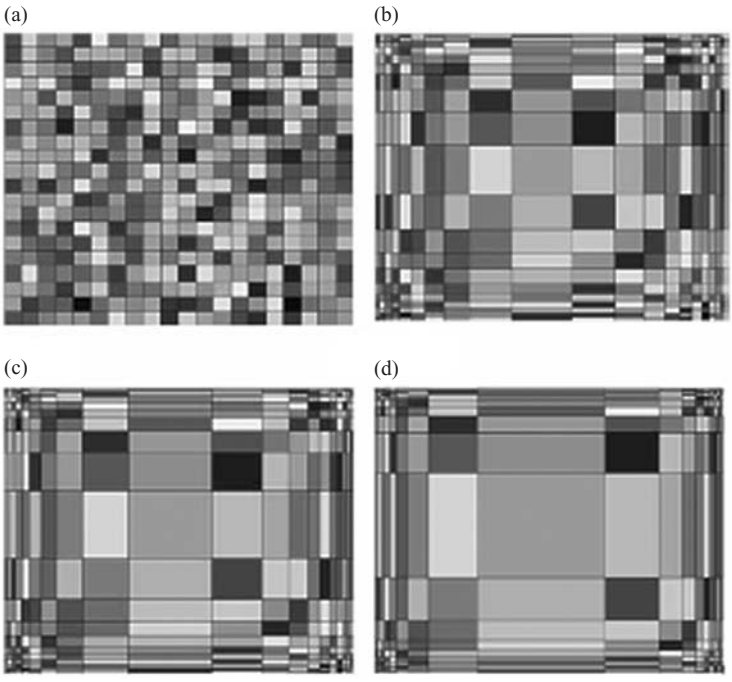

Figure X.30. Fisheye treatments of a SOM mapped onto a 20 × 20 grid with various distortion values; this type of display is commonly used in maps of concepts within categories. (From Yang, Chen, and Hong 2003. Reprinted with permission from Elsevier.)

Filtering Fisheye Views

Filtering fisheye approaches, such as *fractal approaches*, focus on the control of information in the creation of display layouts. Such approaches attempt, through approximation, to create simpler abstractions of complex structures by filtering the amount of information displayed in a way corresponding to some system- or user-defined threshold. Examples of filtering view approaches are found in Figure X.31.

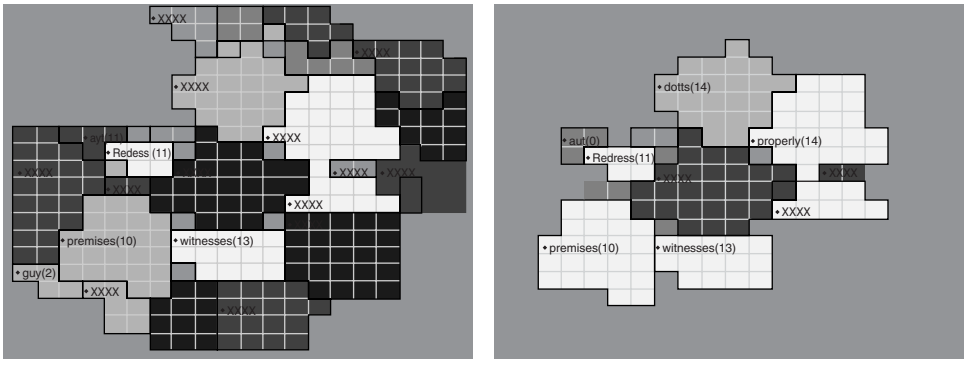

Figure X.31. Filtering view approaches (fractal view) applied to the same category map at different threshold settings. (From Yang, Chen, and Hong 2003. Reprinted with permission from Elsevier.)

Yang, Chen, and Hong (2003) has summarized an approach to creating a fractal view of a category map:

- The fractal dimension of a structure D is the similarity dimension of a structure, which is controlled by a scale factor and a branching factor,

$$D = -\log_{r_x} N_x,$$

  where r_x represents the scale factor and N_x represents the branching factor.
- Solving the fractal requirement requires that the relation between the number of branches and the scale factor at each node of the structure shown below exist:

$$\log_{r_x} N_x = \text{constant}.$$

- Formalizing the fractal views entails taking the focus point into account and regarding it as root. Fractal values are propagated to other nodes based on the following formulation:
  - Fractal value of focus point $= F_{\text{focus}} = 1$.
  - Fractal values of the child of region x in a category map $= F_{\text{child of } x} = r_x F_x$, where F_x is the fractal value of x, $r_x = C \times N_x^{-1/D}$, C is a constant, $0 \leq C \leq 1$, D is the fractal dimension, and N_x is the branching factor.

Control in this type of view is maintained by the setting of the threshold values. Regions of the category map with fractal values below the threshold disappear or become diminished.

Applications to Link Detection and General Effectiveness of Fisheye Approaches

Both distorting and filtering fisheye approaches are particularly useful to link detection operations aimed at performing degree-of-relatedness or degree-of-separation analyses. By being able to maintain a focal point on vertices representing known data, users substantially enhance their ability to identify and explore connections with vertices on the diminished but still viewable periphery of the graph.

Moreover, the ability – supported by many fisheye-type interfaces – to move an item that is on the periphery to the focal point quickly through direct manipulation of graph elements while not completely losing sight of the earlier focused-upon vertex (which will have moved, in turn, to the periphery) can be quite important. Indeed, beyond just generally acting to encourage greater interaction with the text mining system, this type of functionality allows users to sift more confidently through relationship data without a feeling of disorientation or "getting lost."

Distorting and filtering fisheye approaches are not mutually exclusive. When dealing with very large volumes of data, link detection operations aimed at discovering the network of truly *interesting* relationships linked to a known concept can be greatly enhanced by being able both (a) to see as much of a peripheral context as possible (via a distorting view approach) and (b) to winnow the overall display of data by means of the threshold setting (via a filtering view algorithm).

Yang, Chen, and Hong (2003) found that both distorting and filtering view approaches were substantially more effective (speed measure) in helping users discover information versus having no visualization tool at all. Yang et al. also found that users achieved faster discovery results employing filtering view approaches versus distorting view approaches but found that visualizations incorporating both

Figure X.32. Visualization of a category map relying on both distorting view and filtering view techniques. (From Yang, Chen, and Hong 2003. Reprinted with permission from Elsevier.)

distorting view and filtering view functionality were the most effective at increasing the speed of discovering useful data. An example of a visualization incorporating both distorting view and filtering view approaches can be seen in Figure X.32.

X.4.3 Spring-Embedded Network Graphs

Link analysis activities benefit from visualization approaches that offer quick spatial and layout cues to the relative proximity that certain relations between concepts possess. *Spring embedding* is a graph generation technique first described by Eades (and later refined in significant ways by both Kamada and Kawai and Fruchterman and Rheingold) that distributes nodes in a two-dimensional plane with some level of separation while attempting to keep connected nodes closer together relative to some form of weighting scheme. Spring graphs are a common form in many academic and commercial text mining applications with an orientation toward link detection such as ClearForest's *ClearResearch* (see Figure X.33) and Paul Mutton's *PieSpy* social network visualization software (Mutton 2004) (see Figure X.34).

In generating a *spring-embedded network graph*, or *spring graph*, we regard each node as a kind of "charged particle" within a graph model that simulates a closed-force system. This formulation creates a repulsive force between every pair of nodes in the system. Each edge in the graph, on the other hand, is modeled as a spring that applies an attractive force between the pair of nodes it links.

Ultimately, the full spring graph is drawn in iterations that calculate the totality of repulsive and attractive forces acting on nodes within the closed system. At the

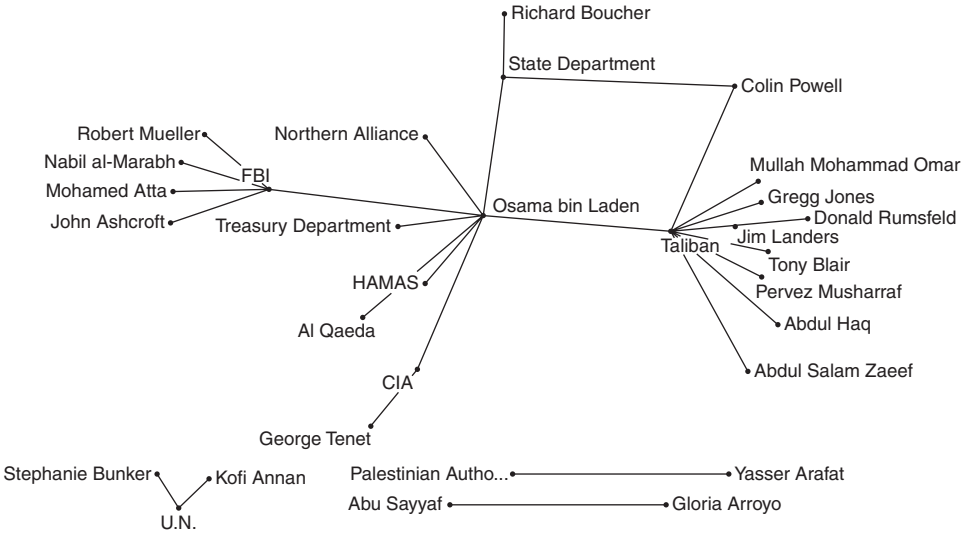

Figure X.33. Spring graph of person concepts associated with organization concepts in the context of terrorism.

close of each iteration, all the nodes in the system are moved according to the forces that were applied during that iteration's calculations.

Typically, in most practical situations, the creation of spring graphs occurs in a multistage process. Running a spring-embedder algorithm is only one stage in

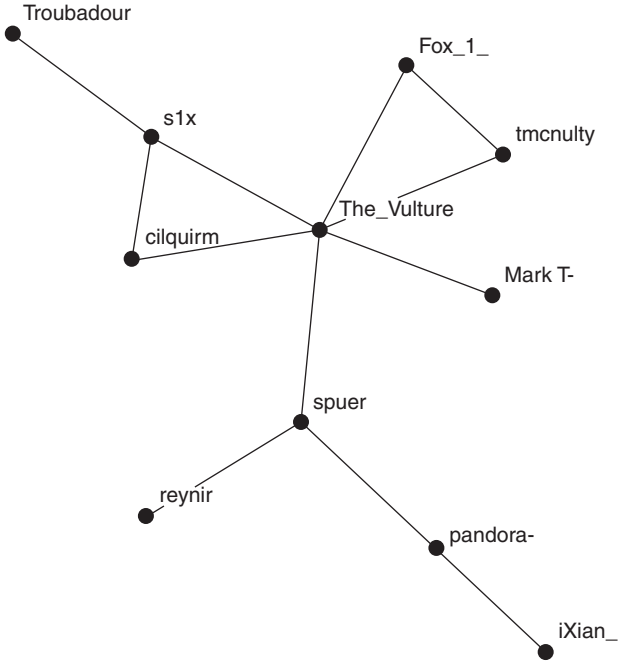

Figure X.34. Simple social network of Internet Relay Chart (IRC) users depicted in a spring graph by the PieSpy social network visualization application. (From Mutton 2004. Reprinted with permission, © 2001 IEEE.)

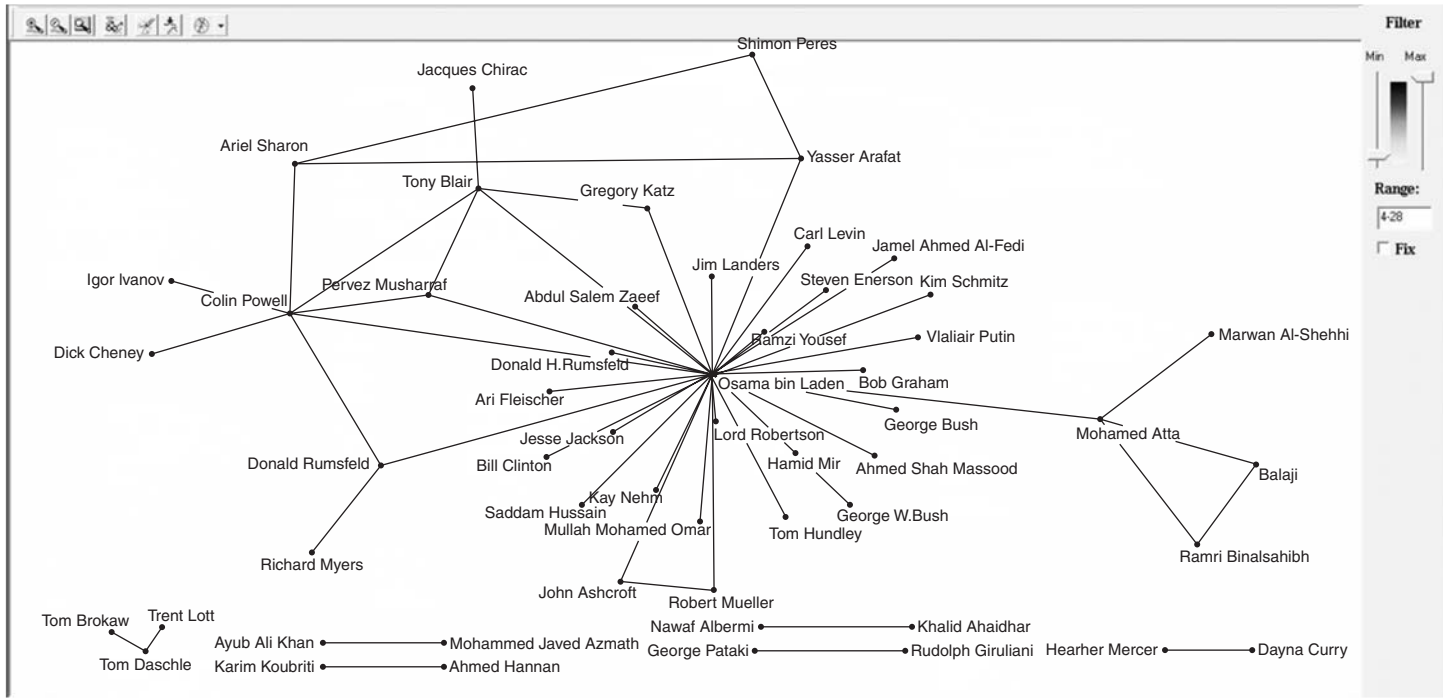

Figure X.35. GUI with visualization of disconnected spring graph showing person co-occurrence patterns.

this process, which would customarily also include some customized preprocessing routines to reduce complexity and heuristics to help establish clusters and perform other processes to promote faster generation of spring graphs in real-time graph-rendering situations. A full example of the construction of a social network spring graph can be found in Chapter XI.

Spring graphs can range in size from a handful of nodes to the hundreds of thousands. Spring graphs whose nodes are all linked by edges are called *connected spring graphs*; those in which discrete networks of nodes appear are referred to as *disconnected spring graphs* (see Figure X.35).

Link detection applications leverage spring graphs to provide visual cues in network maps in which edge length corresponds to the actual relatedness of two nodes. These visual cues allow a user to visually trace out degrees of relatedness and separation quickly, making pattern exploration more effective. Moreover, the spring graphs' ability to model extremely large networks makes them doubly useful in link detection activities involving very large data collections.

X.4.4 Critical Path and Pathway Analysis Graphs

Link analysis can also be visualized through directed graphs that show the linked events or paths of interrelated actions. *Critical path diagrams* are typically based on a graphical model called an activity network, which is a form of DAG. Unlike most DAGs, in which emphasis is usually placed on the vertices of the graph, critical path diagrams equally emphasize the nodes – which typically represent either entities or events – and the edges – which can represent tasks, actions, or decisions. Figure X.36 shows a rudimentary critical path diagram.

In such diagrams, a *critical path* is a chain of specific nodes and edges – or entities events, and the tasks or actions that connect them – that demonstrate some level of interestingness. As in Figure X.36, the patterns formed by such chains of nodes

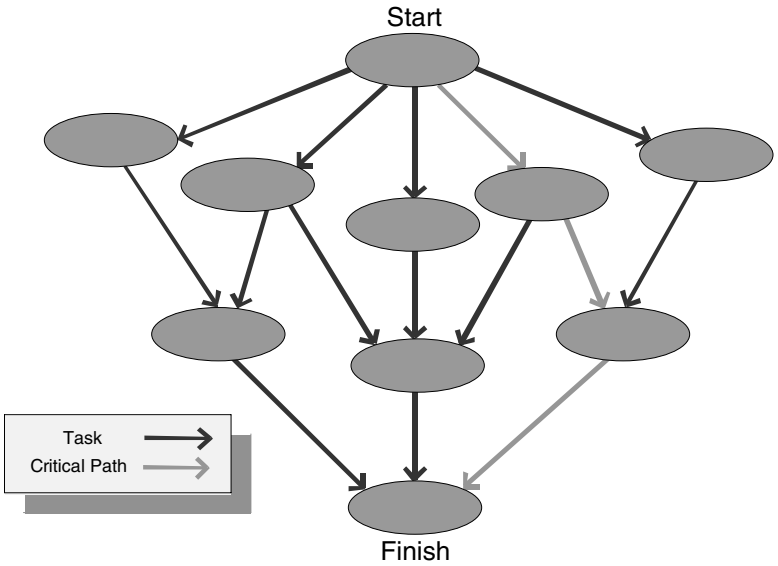

Figure X.36. Critical path diagram.

and edges can be highlighted by stylistic elements in the visualization process (e.g., a different color for edges that link nodes in this chain, etc.). Frequently, though not always, these critical paths will have an identifiable start and finish and thus constitute a directed subgraph that is part of a wider activity network.

Critical path graphs are a staple part of link detection activities aimed at investigations of criminal activities. In crime analysis visualization graphs, nodes may represent both entities (persons, places, items) and events (crimes, pretrial proceedings, trials). Also, a timeline may be introduced to frame actions that occur over time.

Visualizations that support critical path analysis share similarities with the graphic approaches used in pathways analysis for genomics and proteomics research, though there are also some differences. Link detection systems emphasize the search for chains or pathways of interactions between proteins, drugs, and diseases in directed graphs. Edges in these directed graphs are often highlighted in color coding to identify a pathway – but this color coding of edges is also used to specify different types of interactions.

X.4.5 Citations and Notes

Sections X.4–X.4.3

Fisheye views were introduced by G. Furnas; probably the best early description is in Furnas (1986). Subsequently, Sarkar and Brown (1992) added useful upgrades to fisheye views of data and abstracted the general algorithmic approach used to generate fisheye views. The algorithmic formulation for fisheye views comes from Sarkar and Brown (1992). Yang, Chen, and Hong (2003) provides a good treatment of distorting and filtering approaches taken with fisheye views; Noik (1996) also contains a useful discussion.

Figures X.30, X.31, and X.32, as well as the generalized approach to creating a fractal view of a category map discussed in Section VI.4.2, have been summarized from Yang, Chen, and Hong (2003). Yang et al. apply various fisheye approaches to a category map generated using a Kohonen-style SOM. Koike (1995) offers very useful background on the use of fractal approaches as filtering-view techniques.

Sections X.4.4–X.4.5

Spring-embedded network graphs were introduced in Eades (1984) and refined in several subsequent papers – perhaps most notably, Kamada and Kawai (1989) and Fruchterman and Reingold (1991). More on ClearForest's ClearResearch product can be found at <www.clearforest.com>. Further discussion of PieSpy can be found in Mutton and Rodgers (2002) and Mutton (2004).

Mutton and Golbeck (2003) suggests the formulation of a spring graph as a closed-force system in which every node is a "charged particle." The spring graph in Figure X.34 comes from Mutton (2004).

X.5 REAL-WORLD EXAMPLE: THE DOCUMENT EXPLORER SYSTEM

Initially developed in 1997, Document Explorer is a full-featured text mining system that searches for patterns in document collections. Such a collection represents an application domain, and the primary goal of the system is to derive patterns that

provide knowledge about this domain. The derived patterns can be used as the basis for further browsing and exploration of the collection.

Document Explorer searches for patterns that capture relations between concepts in the domain. The patterns that have been verified as interesting are structured and presented in a visual user interface allowing the user to operate on the results, to refine and redirect mining queries, or to access the associated documents. Like many general text mining systems, Document Explorer focuses on the three most common pattern types (e.g., frequent sets, associations, distributions); however, it also supports exploration of textual data by means of keyword graphs.

Perhaps most notably for a real-world system of its time frame, Document Explorer provides a well-rounded suite of complementary browsing and visualization tools to facilitate interactive user exploration of its document collection. Examination of Document Explorer with this in mind can provide useful insights into how a practical text mining system leverages presentation-layer tools.

The Document Explorer system contains three main modules. A diagram of the overall Document Explorer system architecture is shown in Figure X.37. The first module is the backbone of the system and includes the KDTL query front end (see Section II.3), into which the user can enter his or her queries for patterns; the interpreter, which parses a query and translates it into function calls in the lower levels; and the data mining and the data management layer. These two layers are responsible for the actual execution of the user's query. The data mining layer contains all the search and pruning strategies that can be applied for mining patterns. The main patterns offered in the system are frequent concept sets, associations, and distributions.

The embedded search algorithms control the search for specific pattern instances within the target database. This level also includes the refinement methods that filter redundant information and cluster closely related information. The data management layer is responsible for all access to the actual data stored in the target database. This layer isolates the target database from the rest of the system.

The second module performs source preprocessing and categorization functions. This module includes the set of source converters and the text categorization software. It is responsible for converting the information fetched from each of the available sources into a canonical format for tagging each document with the predefined categories, and for extracting all multiword terms from the documents. In this preprocessing component, the system extracts all the information that will subsequently be used by the data mining methods.

The target database is represented as a compressed data structure. Besides the target database, the text mining methods in Document Explorer exploit a knowledge base on the application domain. The terms of this domain are arranged in a DAG and belong to several hierarchically arranged categories. In the Reuters newswire collection used in this example, the main categories correspond to countries, economic topics, persons, and so on. Each category (e.g., economic topics) has, for example, subcategories such as currencies and main economic indicators. Relations between these categories give further background knowledge. The knowledge base for the Reuters collection includes relations between pairs of countries (such as countries with land boundaries), between countries and persons, countries and commodities, and so on. These relations can be defined by the user or transformed by special

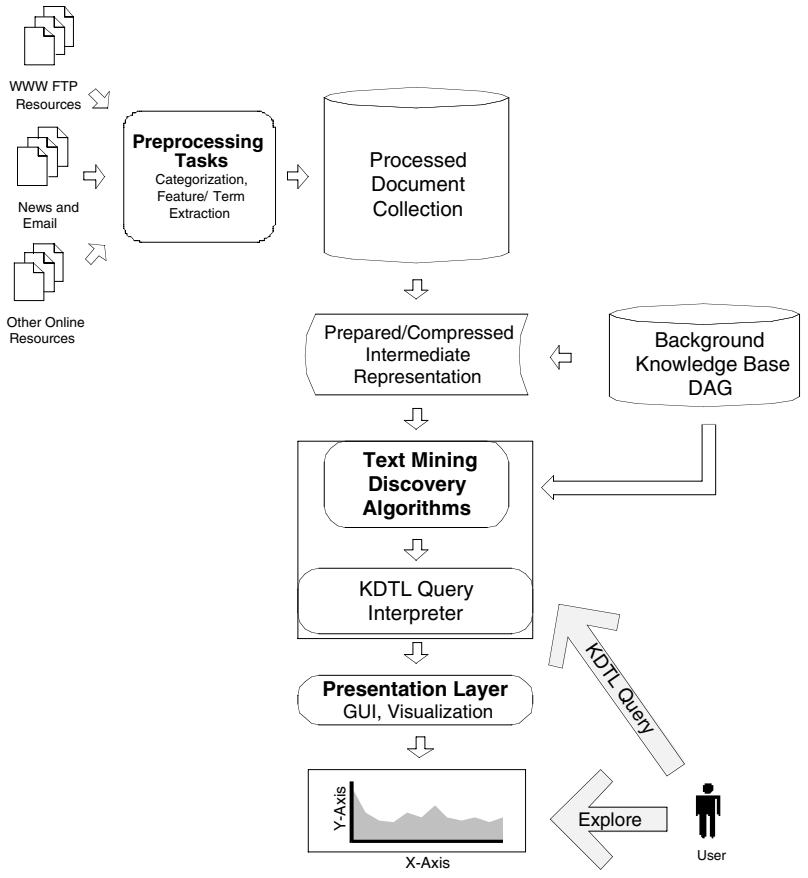

Figure X.37. Architecture of the Document Explorer system.

utilities from generally available sources such as the CIA World Fact Book or companies' home pages.

Finally, the third module performs presentation-layer functions and is responsible for providing an attractive set of GUI-based text mining tools and graph-based visualization techniques that give the user a much easier access to the system. Simple concept graphs are a special interactive visualization technique to present data mining results. Simple concept graphs extend the notion of association rules to relations between keywords and phrases occurring in different documents. The focus of the following functional descriptions is on this presentation layer module.

X.5.1 Presentation-Layer Elements

Visual Administrative Tools: Term Hierarchy Editor

To make full use of Document Explorer's knowledge discovery tools, the documents' annotations are grouped into categories of related terms (e.g. country names, machine parts, etc.) and placed in a hierarchical structure. The *Term-Hierarchy* editor, included in Document Explorer, provides a graphical tool for easy construction and

manipulation of such hierarchies. Document Explorer also comes with a predefined term hierarchy for common topics.

The Knowledge Discovery Toolkit

Document Explorer places extensive visualization and browsing tools at the user's disposal for viewing the results of the discovery process. The user is provided with dynamic browsers, which allow dynamic drill-down and roll-up in order to focus on the relevant results. Any part of the discovery process can either be applied to the entire collection or to any subsets of the collection.

Throughout the mining operation, the system maintains the links to the original documents. Thus, at any stage in the discovery process, the user can always access the actual documents that contributed to the discovered pattern.

Document Explorer tools can be grouped into four main categories: Browsers, Profile Analysis, Clustering, and Pattern Discovery. In addition, the system provides novel visualization techniques.

Browsers

The Document Explorer discovery process starts at the browsing level. Browsing is guided by the actual data at hand, not by fixed, rigid structures.

Document Explorer provides two dynamic, content-based browsers: *distribution browser*, and the *interactive distribution browser*.

- ***Distribution Browser***. The distribution browser presents the user with the frequency of all terms (concepts) in the collections grouped by category and allows the collection to be browsed based on these frequencies. In addition, the user can specify a *base* concept, and the browser will present him or her with the distribution of all other concepts with respect to the base concept. With this tool, the user can immediately find the most relevant term related to whatever he or she is interested in. For example, given a collection of news articles, the user may immediately learn that the main business of Philip Morris is tobacco, or that Wang Yeping is strongly affiliated with China (she is the President's wife). This information is obtained before even reading a single document. At any time, the user may drill down and access the actual documents of interest.
- ***Interactive Distribution Browser***. The interactive distribution browser provides the user with a flexible, interactive browsing facility, allowing him or her to navigate through the data while being guided by the data itself (see Figure X.38). This browser allows the user to zoom in and out on sets of concepts in the collection and obtain online information on the distribution of these concepts within the collection and their relation to other concepts. At any time, the user may drill down and access any document of interest by first clicking on a term in the interactive distribution browser's distribution tree GUI, hitting a button to locate all documents containing the term, and then choosing from a list of titles for these documents to access the full text of the document.

Visualization Tools

Document Explorer is equipped with a suite of visualization tools. These aid the user in gaining a quick understanding of the main features of the collection. The

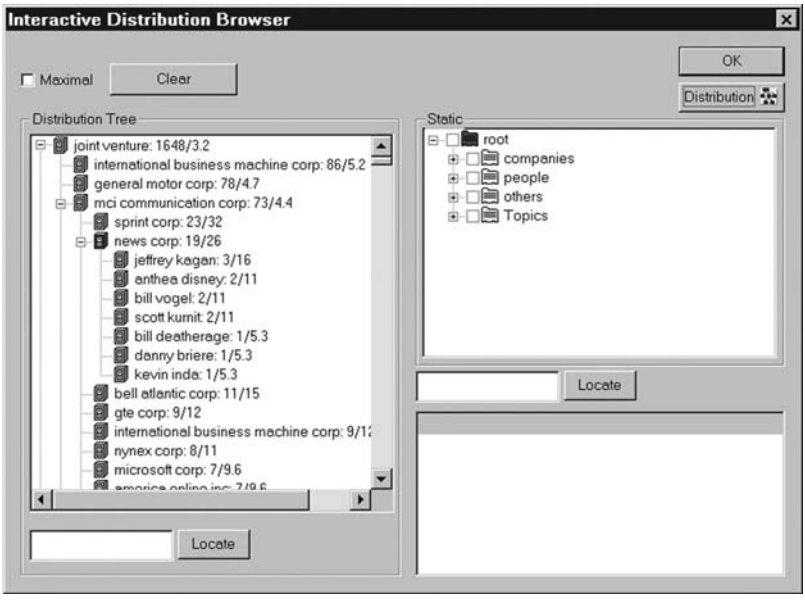

Figure X.38. The GUI for Document Explorer's interactive distribution browser. (From Feldman, Kloesgen, and Zilberstein 1997b.)

visualization tools afford a graphical representation of the connection between terms (concepts) in the collection. The graphical representations provide the user with a high-level, bird's-eye summary of the collection. Three of Document Explorer's main visualization tools – *simple concept graphs, trend graphs*, and *category connection maps* – are described here.

- ***Simple Concept Graphs.*** As described in Section IV.3.1, a simple concept graph in Document Explorer consists of a typical set of graph vertices and edges representing concepts and the affinities between them. A simple concept graph in Document Explorer is generally defined with respect to a *context*, which determines the context in which the similarity of keywords is of interest. Figure X.39 shows a simple concept graph for the "country" category in the context of "crude oil," while Figure X.40 illustates a simple concept association graph with multiple categories but only one vertex.

  In Document Explorer, simple concept graphs can either be defined for the entire collection, or for subsets of the collection, and for arbitrarily complex contexts (see Figure X.41). The system provides the user with an interactive tool for defining and refining the graphs.
- ***Trend Graphs.*** Trend graphs (see Section II.1.5) provide a graphical representation of the *evolution* of the collection. The user is presented with a dynamic picture whose changes reflect the changes in the collection.

  The user can focus on any slice in time and obtain the state of the information at the given time. The user can also define the granularity at which the information is analyzed and presented.
- ***Category Connection Maps.*** This visualization tool enables the user to view the connections between several different categories in relation to a given context.

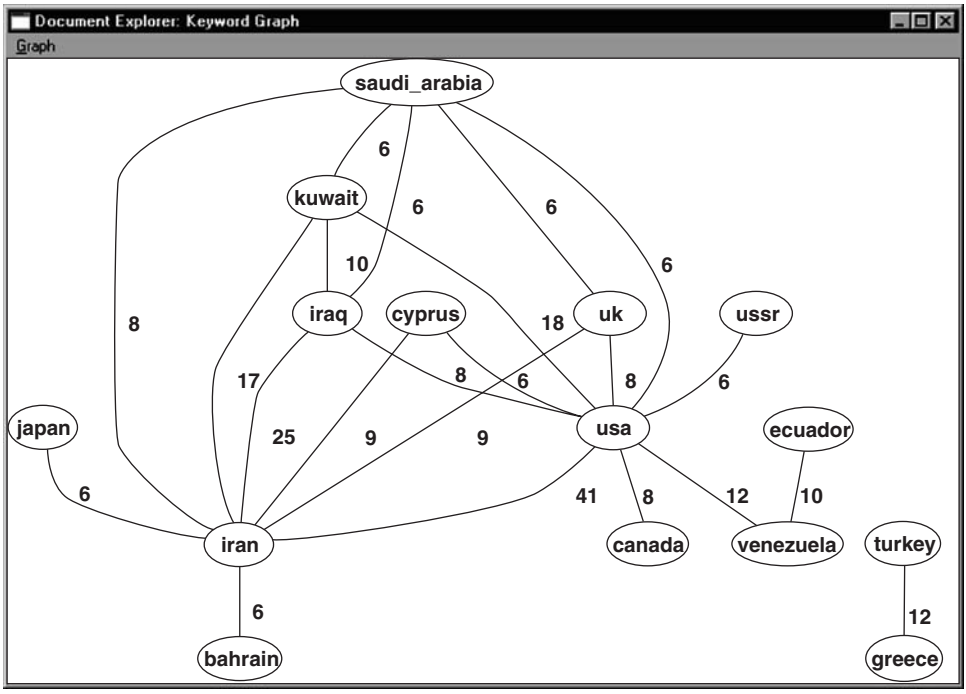

Figure X.39. A Document Explorer simple concept graph – "Countries" in the context of "Crude Oil." (From Feldman, Kloesgen, and Zilberstein 1997b.)

Figure X.42 presents the connections between the categories: *people*, *brokerage houses*, and *computer companies* within the context of *mergers*. (Some similar sample implementations of the circle graph as category connection map are described in Section XII.2.2.)

X.5.2 Citations and Notes

For a comprehensive overview of Document Explorer, see Feldman, Kloesgen, and Zilberstein (1997a, 1997b). The original Document Explorer development team included Ronen Feldman, Yonatan Aumann, David Landau, Orly Lipshtat, Amir Zilberstien, and Moshe Fresko.

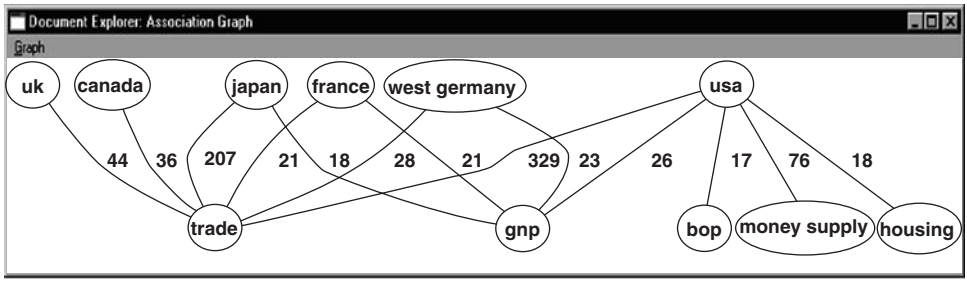

Figure X.40. Simple concept association graph from Document Explorer – many categories but one vertex. (From Feldman, Kloesgen, and Zilberstein 1997b.)

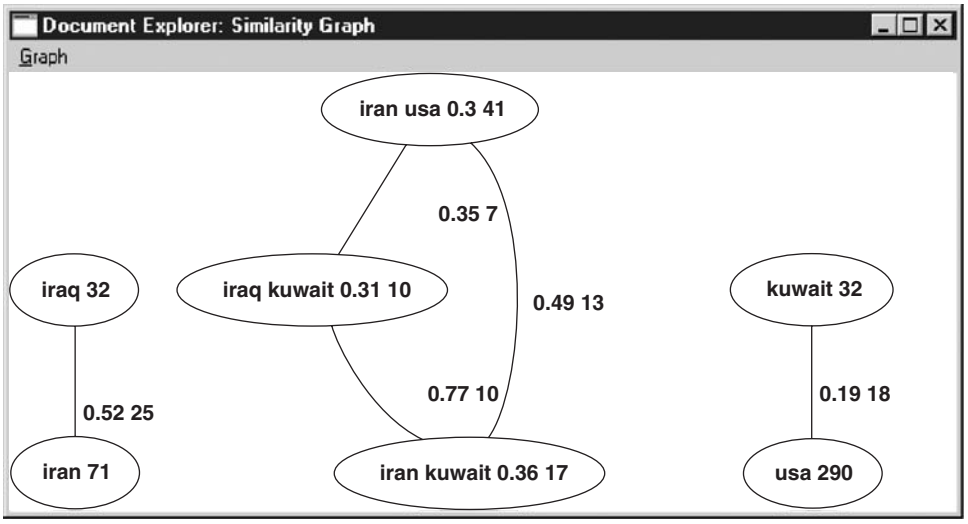

Figure X.41. Simple concept graph from Document Explorer – interesting concept sets and their associations context: crude oil; categories: countries. (From Feldman, Kloesgen, and Zilberstein 1997b.)

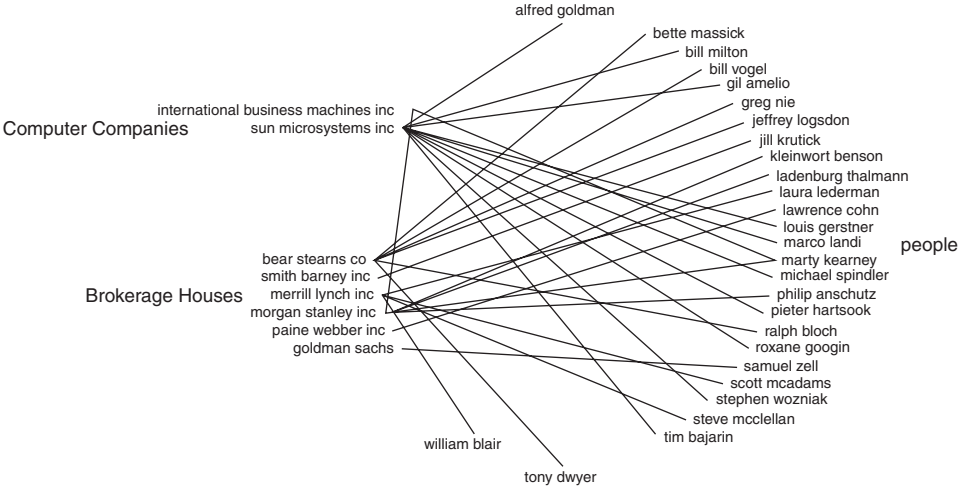

Figure X.42. Category map for "People," "Brokerage Houses," and "Computer Companies" with respect to "Mergers." (From Feldman, Fresko, Hirsh, et al. 1998.)